Object Counting and Localization: A Statistical Approach

by

Purnima Rajan

A dissertation submitted to The Johns Hopkins University
in conformity with the requirements for the degree of
Doctor of Philosophy

Baltimore, Maryland
September, 2018

Abstract

Scene understanding is fundamental to many computer vision applications such as autonomous driving, robot navigation and human-machine interaction; visual object counting and localization are important building blocks of scene understanding. In this dissertation, we present: (1) a framework that employs doubly stochastic Poisson (Cox) processes to estimate the number of instances of an object in an image and (2) a Bayesian model that localizes multiple instances of an object using counts from image sub-regions.

Poisson processes are well-suited for modeling events that occur randomly in space, such as the location of objects in an image or the enumeration of objects in a scene. The proposed algorithm selects a subset of bounding boxes in the image domain, then queries them for the presence of the object of interest by running a pre-trained convolutional neural net (CNN) classifier. The resulting observations are then aggregated, and a posterior distribution over the intensity of a Cox process is computed. This intensity function is summed up, providing an estimator of the number of instances of the object over the entire image. Despite the flexibility and versatility of Poisson processes, their application to large datasets is limited, as their computational complexity and storage requirements do not easily scale with image size,

typically requiring $O(n^3)$ computation time and $O(n^2)$ storage, where n is the number of observations. To mitigate this problem, we employ the Kronecker algebra, which takes advantage of the tensor product structure of covariance matrices. As the likelihood is non-Gaussian, the Laplace approximation is used for inference, employing the conjugate gradient and Newton's method. Our approach has then close to linear performance, requiring only $O(n^{3/2})$ computation time and $O(n)$ memory. We demonstrate the counting results on both simulated data and real-world datasets, comparing the results with state-of-the-art counting methods.

We then extend this framework by noting that most object detection and classification systems rely upon the use of region proposal networks or upon classifying the "objectness" of specific sub-windows to help detect potential object locations within an image. We use our Cox model to convert such region proposals to a well-defined Poisson intensity. This output can be used as-is to directly estimate object counts, or can be plugged into pre-existing object detection frameworks to improve their counting and detection performance. This remapping does not require the original network to be re-trained: the parameters of the model can be estimated analytically from the training data.

Furthermore, we consider the problem of quickly localizing multiple instances of an object by asking questions of the form "How many instances are there in this set?", while obtaining noisy answers. We evaluate the performance of the partitioning *policy* using the expected entropy of the posterior distribution after a fixed number of questions with noisy answers. We derive a lower bound for the value of this problem and study a specific policy, named

the *dyadic policy*. We show that this policy achieves a value which is no more than twice this lower bound when answers are noise-free, and show a more general constant factor approximation guarantee for the noisy setting. We present an empirical evaluation of this policy on simulated data for the problem of detecting multiple instances of the same object in an image. Finally, we present experiments on localizing multiple objects simultaneously on real images.

Thesis Committee

Primary Readers

Bruno M. Jedynak (Advisor)
Maseeh Professor of Mathematical Sciences
Department of Mathematics and Statistics
Portland State University

Gregory D. Hager (Advisor)
Mandell Bellmore Professor
Department of Computer Science
Director, Malone Center for Engineering in Healthcare
Johns Hopkins Whiting School of Engineering

Philippe Burlina
Associate Research Professor
Department of Computer Science
Johns Hopkins Whiting School of Engineering, and
Principal Staff Scientist
Johns Hopkins Applied Physics Laboratory

Acknowledgments

First of all, I would like to thank my advisors Bruno Jedynak and Gregory Hager for the mentoring they have provided me throughout my graduate studies.

Bruno introduced me to statistics and helped me discover non-traditional, creative ways of tackling vision problems. I am grateful for the immense amount of time he has spent teaching me, for being available whenever I needed help and guidance, invariably coming up with new and inspiring ideas.

My co-advisor Greg always encouraged me to take a step back and look at the big picture rather than getting lost in the minute details. His intuition and insights in the field were invaluable when tying up the final stages of my dissertation, and helped extend the work beyond the initial conception.

Professor Phil Burlina took the time to be on the committee and offered valuable inputs on my thesis. Professor Simon Leonard gave me teaching assistantship spanning many semesters, making my graduate life easier than it would otherwise have been; he introduced me to robotics and showed me the geeky, hands-on way of doing things.

It was a great pleasure to work with my collaborators: Yongming Ma,

Weidong Han, Prof. Raphael Sznitman and Prof. Peter Frazier. I also thank my LCSR lab mates Ehsan Azimi, Amir Farvardin, Robert Grupp, Ali Ebrahimi, Preetham Chalasani, Mohit Singhala, Ayushi Sinha and Sharmishtaa Seshamani for fun, helpful and insightful conversations.

I would like to express my gratitude to my friends Hari, Shyam and Kunal for sharing their Ph.D. life with me - exchanging graduate life stories over tea and, later, aiding the rehearsal and development of my thesis defense. My friends and teachers from the university rec center - Charlotte, Claire, Dimitri, Ann, Amber and Ted - also did much to help smooth my journey. Meanwhile, Prof. Julie Reiser helped me change my mindset on professional presentations; her encouragement and reading recommendations were incredibly useful during my last year.

In particular, I would like to highlight how much the support of my family has meant during this period. My parents and brother have always offered me their unconditional encouragement, trust and love, and helped me stay focused on my research.

Finally I would like to thank George who has been with me every step of the way over the last three years, for making me laugh when things get rough, and for having been a constant source of support and inspiration.

Dedication

This dissertation is dedicated to my loving parents Vinodini and Rajan, and my brother Nithin.

Table of Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

One of the most fascinating properties of the human perceptual system is its innate capacity for detecting objects and estimating their number by merely glancing at a scene. Studies have found [1, 2] that human subjects quickly come up with the right number if the count is small, and provide close estimations when the number is higher. However, in either case it is clear that the human visual system perceives objects in scenes as distinct items, at a fundamental level.

This capability, which has also been observed in the animal world [3, 4, 5], suggests an important distinction: there is a difference between merely *encountering* a scene as a visual pattern, and *apprehending* it as a panorama of distinct and meaningful entities. The ability to replicate this in some fashion – that is, to automatically count and locate everyday objects in images of natural scenes – would present a significant step towards a more advanced form of artificial intelligence.

The fundamental building blocks of most computer vision systems are: object detection, localization and counting. Object detection and localization

have already become a very active area of research given the numerous obvious applications, such as human-computer interaction, robotics, autonomous driving, medical imaging, consumer electronics and security surveillance. Object counting, meanwhile, has so far received less attention, perhaps due to the more nuanced applications for which it is suited. Taken together, we arrive at the prospect of an artificial intelligence which knows objects as *objects as part of a spatial grouping*, and therefore acts more appropriately to its broader environment, in an immediate way that is derived directly from its context. In this dissertation, we adopt a Bayesian perspective to address the problems of counting and localization of objects within an image data context, and from there extend upon these ideas.

Object counting has interesting applications in areas such as remote sensing, biology, astronomy, environmental conservation and industrial manufacturing. A typical application in biology is cell counting in microscopic images. This is a subset of *cytometry*, the quantitative analysis of cells and cell systems. In the field of neuroscience, estimating the number of vesicles, synapses or mitochondria from electron microscopy [6] image volumes is important in the work towards understanding the brain processes. Automated counting has applications in environmental survey: for wildlife census collection [7, 8]; or for counting the number of trees in aerial images of forests [9] and urban landscapes [10] with the goal of conservation [11], population management or for measuring the impact of climate change.

Most of the early work in counting was intended for crowd and pedestrian analysis [12, 13, 14]. In very dense crowd counting, the crowd itself is often

treated as a texture; features are computed directly from the crowd instead of detecting individuals that make up the crowd [15]. When the number is moderate and the individual instances are discernible, techniques that take advantage of this fact are used, such as assigning shape priors for the human torso [16] or employing head and face detectors [17]. Applications include estimating the number of pedestrians [18] and cars [19] for traffic signal management and for the modification and expansion of traffic facilities like tunnels and flyovers; and automated estimation of occupancy and movement for the design and analysis of buildings and public spaces [20].

Even in applications where enumeration is not the end goal, the object counts provide a high level feature which could help better understand a scene. Moreover, counting over sub-images allows for shallow localization and even precise localization when the sub-images provide a fine-grained partition of the original image.

"Faster RCNN" [21] is currently one of the best algorithms for object detection on popular datasets like Pascal VOC [22], ILSVRC [23] and MS COCO [24]. However, this algorithm still performs well under human performances for the task of counting on datasets like MS COCO. Current object detection methods use either region proposals [25] or sliding windows at multiple scales [26] to generate the initial set of candidate bounding boxes. An object classifier is run on these proposed boxes, the classification results are further refined by techniques like greedy non-maximal suppression and bounding box regression. Once the objects are detected accurately, counting them is trivial. However, in practice the post processing steps in object detection often require

elaborate fine tuning and optimization.

The problem of localizing structures of interest, or *targets*, appears in numerous applications, such as finding quasars in astronomical data [27], localizing faces in images [28] or counting synapses in microscopy volumes [6]. Once a competent detection scheme is available, the localization task often reduces to evaluating each possible location in an exhaustive fashion. Such strategies are highly effective and easy to implement, which contributes to their widespread use.

Yet such localization strategies do not scale with data size requirements since their computational complexity depends directly on the searchable domain's size. This problem is critical in analysis of enormous microscopy volumes where localizing and counting intra-cellular structures such mitochondria or synapses is critical to understanding brain processes [29]. Thus, efficient localization of object instances in images remains challenging given the growing amount of image data to evaluate. We examine strategies for efficient localization in the second half II of this thesis.

1.1 Thesis Statement

We hypothesize that statistical modeling can be used to advance the state-of-the-art in counting and localization of objects in images of natural scenes. Moreover, a two-part modeling would allow for the separation of the object class from the specifics of the model formulation, resulting in a generic system that is independent of the characteristics of the object of interest.

1.2 Contributions

Part I: Counting

We make the following five contributions in this portion [30] of the work: First, we re-formulate the problem of visual object counting as a Bayesian optimization problem using Cox processes; second, we incorporate Kronecker algebra into this model for efficiency; third, we derive analytical expressions for the hyper-parameters using the *method of moments*; fourth, we present an efficient algorithm for the computation of the posterior distribution that facilitates counting; and finally, we enhance region proposal networks using our Cox model so as to improve the performance of pre-existing counting and detection networks.

Part II: Localization

In this section [31, 32], (i) we propose and analyze the dyadic policy for simultaneous localization of multiple instances of an object in image data context. We make use of object counts from image sub-regions to eventually localize them; and dyadic policy is one of the strategies for partitioning the image search space. This policy can be computed quickly and is non-adaptive, making it easy to parallelize, and is far simpler to implement than dynamic strategies. This policy and our analysis uses an observational model that allows noise. (ii) We derive an explicit closed form expression for the posterior distribution under the dyadic policy for object localization. This allows easy and exact computation of the expected number of targets at each location in

our search space. (iii) Finally, we provide two algorithms for object localization based on the dyadic policy.

1.3 Dissertation Outline

The rest of the thesis is divided into two parts: I) *Object Counting* and II) *Localization*.

Part I starts with an introduction to Gaussian processes and Cox processes, discussing the basic formulation of such models in a general sense. This is followed by an overview of our counting system and the specifics of the algorithm in chapter 2. Efficiency considerations are discussed in chapter 3. Chapter 4 discusses our approach to hyper-parameter tuning using the *method of moments*. We follow up with simulations and experiments on real image data in chapter 5. Chapter 6 discusses enhancement of region proposal networks such as "Faster RCNN" using Cox models.

Part II switches focus to localization via sub-region counting: chapter 7 includes an introduction and theoretical formalization of the dyadic policy for partitioning the image search space, followed by an explicit characterization of the dyadic posterior distribution. Chapter 8 discusses the algorithms used for object localization based on the dyadic policy in both noisy and noiseless setting. This is followed by experiments on real and simulated data. Towards the end of the same chapter, we discuss a potential analogy of object localization to coding theory, providing an illustrative experiment to support the idea.

Finally, we consolidate the two parts by using the results from the Cox

counter as the input for object localization using the dyadic policy. The results of this experiment is presented in chapter 9.

Part I

Visual Object Counting

Chapter 2

Cox Processes for Counting by Detection

In this part of the dissertation, we reformulate the problem of object counting as a Bayesian estimation problem, specifically by using doubly stochastic Poisson (Cox) processes. Poisson processes [33] are one of the simplest probabilistic models that describe the position of instances of a given object within an image. A (non uniform) Poisson process is fully characterized by a positive function over the image domain called an intensity. According to the definition of a Poisson process, the sum of this intensity over a sub-image provides the expected value as well as the variance of the number of instances within this sub-image. Furthermore, the total number of instances of the object in the image can be estimated by adding the intensity over the whole image domain. In practice, this intensity is not available but can be estimated from data, thus providing an algorithmic solution to the counting problem.

One of the key issues, then, is the generation of the data necessary for a good estimation of the intensity function. How can this best be achieved? In this work, we propose to leverage recent advances in object classification by

running a CNN binary classifier for the object of interest over a dense pixel grid. Despite the overall good quality of the CNN output, visual inspection of the resulting images shows that there are multiple responses per instance, that these "blobs" of responses might be intersecting, of different sizes and corrupted by noise such that the problem of estimating the intensity from this data is not trivial. Note that a crude way of estimating the count directly from the CNN detection output would be to detect the connected components, or count the number of local maxima, but this involves ad-hoc smoothing and thresholding of the classifier output, the parameters of which can only be computed empirically; they also cannot be easily transferred from one object type to the other or from one dataset to another. Note that there exists many sophisticated methods ([34], [35]) with efficient global regularization that develop density estimates from saliency map. Our work aims to provide an alternate approach to estimating this density map from the CNN detection output using Cox modeling.

Since regularization is needed, we adopt a Bayesian point of view. A positive function, called the intensity is used to describe the expected number of instances of an object within a surface element of the image domain. We model the intensity as a random positive function. A Gaussian process (GP) [36] provides a random function, and a link function maps the GP into a probability distribution over intensities. This construction is standard, dating back to the Cox process, introduced by David R. Cox in 1955 [37]. However, the applications are recent due to the relatively high computational burden for computing the posterior and estimating the hyper parameters. These

applications include reinsurance pricing, portfolio optimization [38], and estimating crime maps in Chicago [39]. The application to visual counting is new to the best of our knowledge.

Inference with Cox processes is complicated since there is no conjugated model. Instead, an approximate method must be used, the simplest of which is the Laplace method [40]. It involves approximating the posterior distribution over the GP with a GP. The posterior mean is obtained by solving a convex minimization problem using the Newton method. Still, this method cannot be used directly because it scales poorly with the dimension of the image. The same problem occurs in GP classification [36] as well, where the computational and storage costs are $O(n^3)$ and $O(n^2)$ respectively. Saatçi [41] presents a technique for using Kronecker algebra which makes the GP inference more tractable, with $O(n^{3/2})$ measurements and $O(n)$ memory requirements without any loss of accuracy. This method requires that the input locations remain on a lattice, but this is not a limitation for computer vision applications as this is a natural structure for images.

We make the following three contributions in this part of the thesis: first, we reformulate the problem of visual counting using Cox processes; second, we incorporate Kronecker algebra into this model for efficiency; and finally, we present an efficient algorithm for the computation of the posterior distribution that facilitates counting.

The remainder of this part of the thesis is organized as follows. In Sect. 2.1, we present a survey of the related work. This is followed by a brief introduction to Cox processes and Gaussian processes Sect. 2.2. A detailed

description of our algorithm and methodology follows from Sect. 2.3 until the end of this chapter. We outline strategies for performance optimization and hyperparameter tuning in chapters 3 and 4. Chapter 5 demonstrates the performance of the proposed algorithm on real and simulated data, and chapter 6 outlines how our Cox model can be used to improve region proposal networks.

2.1 Related Work

Most of the previous work in object counting falls into roughly three categories, (1) counting by density estimation, (2) counting by regression and (3) counting by detection.

Counting by density estimation: In this class of solutions, the counting problem is reformulated as the task of estimating an image density, the integral of which provides the count of objects in the image. These methods [13], [15], [42], [43], [44] typically learn a mapping between local image features and object density, which allows the estimation of a density map for new unseen images. In [42], the density is a linear function of a feature vector associated with each pixel, which is estimated by minimizing a quadratic cost function. In contrast, the work in [43] uses a regression random forest that learns the mapping between image patch features and patch density. The image density is then obtained by averaging over patch-wise density predictions. Most of these methods use domain-specific visual features based on SIFT [45] and HOG [46] as these were proposed prior to the widespread use of CNN features. The accuracy of these methods depend significantly on the choice of image

features. [44] proposes an interactive counting strategy where the framework learns from annotated regions of the image and computes a density map for the non-annotated regions in the same image, allowing the user to inspect the results and refine the estimations. These algorithms, having primarily been developed for crowd analysis, are generally untested outside this domain.

Counting by regression: Here, the object count is estimated by mapping from a set of global features to the integer counts, instead of estimating the count by integrating a density function. In [47] blob size histograms and edge orientations are used as features for estimating the number of pedestrians in an image. [48] and [49] use edge features and texture information based on grey-level transition probabilities in order to directly estimate crowd density using neural nets. These approaches typically discard information about the location of objects, using only the total count for learning.

Counting by detection: In this category of solutions, it is assumed that there is a visual object detector that is tuned to find individual instances of the object. Once the instances are localized, counting becomes a trivial task. Classification networks like *Alexnet* [50] and *VGGNet* [51] have the ability to classify images, but are limited by the fact that they require a fixed-size input image, which means that the image sub-regions should be made to fit either via cropping or warping, leading to distortion. *SPP-Net*[52] solved this problem by using a convolutional feature map from the entire image and then pooling features in arbitrary sub-regions of the image to generate the fixed length representation required in the later layers. The more recent YOLO [53] [54] reframes detection as a regression problem, using a single convolutional network to predict a set

of bounding boxes and class probabilities. *Faster-RCNN* [21] built up on their earlier work [55] [56] [52] for object detection with region proposal networks, and localize objects with a *mAP*(mean average precision) of 42.7% on the MS COCO dataset. When applied to counting *birds* in the same dataset, *Faster-RCNN* is found to have a root mean square error (RMSE) of around 2 (see Table 5.1). The error observed in this result and for other object types suggests that there is scope for further improvement.

In addition to the above categories, there have been many recent interesting work in counting using Bayesian modeling. Pham et al. [57] employs point process inference for large scale object detection and counting, while in [16], a Bayesian marked point process is developed to detect and count people in crowded scenes, leading to an estimate of the count, location and pose of each person in the scene. Point processes allow convenient modeling and analysis of spatial data, the object configuration and the interaction between objects. Marked point processes extend point processes by adding specific marks that associate a parametric object to each point. [58], [59] and [35] use models that allow the representation of images in terms of simple geometric features, with the goal of estimating counts.

CNN-based counting approaches offer powerful improvements over methods that rely on hand-crafted representations [60]. Wang et al. [61] developed an end-to-end CNN regression model for counting people in images of extremely dense crowds. Walach and Wolf [62] learn a density map estimated directly from the input image employing layered boosting and selective sampling. Sindagi and Patel [63],on the other hand, use a cascaded network of

CNNs to jointly learn crowd count classification and density map estimation in densely crowded scenes. [64] propose a switching CNN that leverages intra-image crowd density variation to improve crowd count estimates. Rubio and Sastre [65] developed a Counting CNN where the network learns to map the appearance of image patches to their object density maps.

The proposed algorithm uses a hybrid approach, leveraging the benefits of both *counting by detection* and *counting by density estimation*. We use an initial set of detections as input measurements to our Cox model. We then estimate a posterior density over the entire image using these detection results. The integral of this density provides an estimate of the expected number of objects in the image. It follows that the object count in image subregions can be estimated by integrating the posterior intensity over that subregion.

2.2 Gaussian Processes and Cox Processes

2.2.1 Gaussian Process

Regression is the process of estimating a function $f(\mathbf{x})$ that maps the input data $\mathbf{x}$ to the output y such that continuous real valued predictions can be made on new unobserved data $\mathbf{x}_*$. In traditional regression, the parameters of the function - be it linear or non-linear - are estimated by some form of loss minimization on the training data. A Gaussian process (GP), in contrast, is a non-parametric stochastic process that describes a distribution over functions. The GP defines a prior probability on all possible functions, which when combined with the observed data gives rise to the posterior distribution over functions.

More formally, a Gaussian process (GP) [36], is defined as a collection of random variables, any finite number of which have a joint Gaussian distribution. A Gaussian process can be written as, $g(x) \sim GP(\bar{g}(x), k(x, x'))$, and is completely specified by its mean function $\bar{g}(x)$ and covariance function $k(x, x')$, where $k(x, x')$ is a positive definite kernel. We say that $g \sim GP(\bar{g}, k)$ is a GP when for any collection of points on the input space $(x_1, \ldots, x_n)$, the vector of real numbers $(g(x_1), \ldots, g(x_n))$ is distributed as a multivariate Gaussian with mean $\bar{g}$ and covariance Σ, such that $\Sigma_{ij} = k(x_i, x_j)$.

A positive definite kernel is a function of two arguments such that the resulting matrix Σ is a covariance matrix. The marginalization property of the GP follows directly from this specification of the covariance matrix. This property means that if the GP specifies $(g(x_1), g(x_2)) \sim \mathcal{N}(\bar{g}, \Sigma)$, then it must also specify $g(x_1) \sim \mathcal{N}(\bar{g}_1, \Sigma_{11})$, where Σ_{11} is the relevant sub-matrix of Σ. The properties of the covariance function determines the smoothness of the prior functions. Since GP is non-parametric, it has no notion of fitting the data, and the learning in GP is the problem of finding the appropriate type of covariance function and its corresponding parameters.

Following the notation in *Rasmussen and Williams* [36], given a dataset D of n observations, $D = \{(\mathbf{x}_i, y_i) | i = 1, \ldots, n\} = (X, \mathbf{y})$, we assume that the relationship between the input data X and the target $\mathbf{y}$ is governed by a latent GP function $g(x) \sim GP(\bar{g}, k)$, and $p(y(x)|g(x))$ is the observation model or the likelihood. In Gaussian Process regression, the goal is to find a predictive distribution $p(g_*|y, X, X_*)$ for any new test input set X_*. The prior distribution over functions is Gaussian, and the joint prior distribution of the

training outputs and the test outputs is Gaussian as well. To get a posterior distribution over functions, the joint prior distribution is restricted to contain only those functions that agree with the training data. This is achieved by conditioning the joint prior on the observed data points. The conditional posterior distribution is also Gaussian, and the posterior mean and covariance can be evaluated analytically, following the properties of multi-variate normal distributions, as demonstrated in [36]. The function values corresponding to the test inputs can be then sampled from this posterior distribution.

In the standard GP regression formulation, the observation model is Gaussian, which when combined with a Gaussian prior results in a Gaussian posterior, and the posterior distribution remains analytically tractable. However, for most other applications of Gaussian Processes, including GP classification, the solution is more demanding since the likelihood is typically non-Gaussian.

2.2.2 Cox Processes

Cox processes are also called *mixed Poisson processes* or *doubly stochastic Poisson processes*. A stochastic process [66] $\mathcal{X} = \{X(t), t \in T\}$ is a collection of random variables defined on a common probability space $(\Omega, \mathcal{F}, \mathcal{P})$, where Ω is the sample space, $\mathcal{F}$ is a σ-algebra of subsets of Ω, and $\mathcal{P}$ is a non-negative probability measure on $(\Omega, \mathcal{F})$ with total mass $\mathcal{P}(\Omega) = 1$. For each t in the set T, $X(t)$ is a random variable that represents the state of the process at index t, and t is often interpreted as either time or space. The simplest of stochastic processes is a Bernoulli process, which is a sequence of independent and identically distributed random variables, each of which can take a value

of zero or one based on probability p and $1 - p$ respectively.

Poisson Processes are stochastic processes for collections of points on a domain, the number of points in this collection being also random. A Poisson Process is characterized by a rate or intensity function λ. If λ is constant over the domain Ω, the process is said to be stationary or homogeneous, and if $\lambda(t)$ varies with time or space, the process is inhomogeneous. In a doubly stochastic process, the observed random variables are modeled in two steps: in the first step, the random variables are defined using a stochastic process characterized by one or more parameters, and in the second step, the parameters themselves are treated as random variables. A Cox process, also known as a doubly stochastic Poisson Process is a stochastic process which is a generalization of a Poisson Process where the time(or space)-dependent intensity $\lambda(t)$ is itself a stochastic process. In the case of a Gaussian Cox Process, this intensity is obtained by mapping a Gaussian Process to a positive function using a link function. Examples of link functions include the square, the exponential, the sigmoid, and the logit function. Conditional on the intensity λ, the number of instances within the region Ω is,

$$N(\Omega)|\lambda \sim Poisson\left(\int_{\Omega} \lambda(s)ds\right) \tag{2.1}$$

where $Poisson(u)$ stands for the Poisson distribution with parameter u; and $\lambda(.) = \alpha\phi(g(.))$, where $\phi(g)$ is a positive function and α is a positive scale factor. The function ϕ used here is called is a link function. Link functions are the main ingredients of the Generalized Linear Model. The choice of link functions depends on the details of the optimization procedure and the

specifics of the application, so as to ensure convexity and fast convergence. In our case, we have to choose a link function ϕ such that $\phi \geq 0$, the Hessian matrix of the forward model is negative definite and ϕ^{-1} is lower bounded.

2.3 The Counting Framework

The block diagram in Figure 6.1 provides an overview of the process involved in estimating the number of instances of the object of interest within an input image.

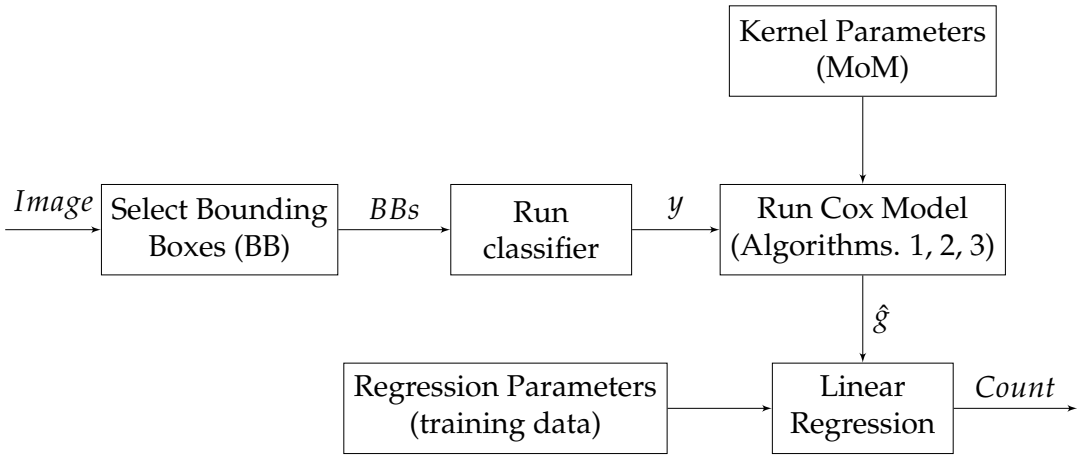


Figure 2.1: The counting work-flow.

First, we select a set of bounding boxes from the image, on which we run a CNN classifier that is trained to classify each bounding box for the presence or absence of the object category that we are interested in. The scaled classifier scores from this initial set of bounding boxes are the measurements y used in the sections that follow. In our Cox modeling, these observations y are functionally related to the intensity of the Poisson process. (The details of this

formulation are described in sections 2.2.2 and 2.4, and the likelihood model is elaborated in section 2.5). The posterior distribution given the observations y is is expressed using Bayes' formula, and the posterior mean $\hat{g}$ is computed using Algorithms 1, 2 and 3 (as detailed in Sects. 2.6, 3.1 and 3.5). Finally, the integral of the posterior mean $\hat{g}$ is mapped using linear regression to find the final estimated count.

2.4 Problem Formulation

Let θ be a doubly stochastic Poisson Process (PP) with intensity λ: $\theta \sim PP(\lambda)$ over the domain $\Omega = [0,1]^d$. In our experiments, $d = 2$. The random intensity function λ is obtained by mapping a Gaussian Process (GP) defined over Ω to a positive function. Let $g \sim GP(\bar{g}, \Sigma)$, where $\bar{g}$ is a function $\Omega \mapsto \mathbb{R}$, and Σ is a positive definite kernel over $\Omega \times \Omega$. We define $\lambda(.) = \alpha\phi(g(.))$, where ϕ is a function $\mathbb{R} \mapsto \mathbb{R}^+$. Examples include $\phi(g) = e^g$ and $\phi(g) = g^2$. $\alpha > 0$ is a scaling factor that together with $\bar{g}$ and Σ control the expected number of counts in the domain.

Consider now a finite grid $\mathcal{D}$ defined over the continuous domain Ω. Moreover, assume that $\mathcal{D} = \mathcal{D}_1 \times \ldots \times \mathcal{D}_d$. That is, $\mathcal{D}$ is the cross product of d one-dimensional grids. This is key to ensuring the scalability of the model. An example for $\mathcal{D}$ when $d = 2$ entails choosing the centers of the pixels of a digital image.

This grid is chosen such that measurements are taken for each of the points in $\mathcal{D}$, or a subset thereof. Notate n, the size of $\mathcal{D}$, and $\tilde{n}$, with $\tilde{n} \leq n$, the number of observed measurements. The locations of the measurements are

$x = (x_1, \ldots, x_{\tilde{n}})$, where $x_m \in \mathcal{D}$. The measurements themselves are notated $y = (y_1, \ldots, y_{\tilde{n}})$. There is no restriction on measurement type. These could be actual counts; or real-valued or vector-valued measurements from one or more classifiers.

Using the Bayes' formula, the posterior on g given the observations y is,

$$\ln p(g|y) \propto \ln p(g) + \ln p(y|g) \tag{2.2}$$

2.5 The Forward Model

In order to define a probabilistic model for y given g, let us define $g_m = g(x_m)$ and $\lambda_m = \alpha\phi(g_m)$, $1 \leq m \leq \tilde{n}$. In the standard Cox process, the observation y_m at x_m is a sample from a Poisson distribution with mean λ_m, for $1 \leq m \leq \tilde{n}$. In visual counting however, y_m is a number in the range $[0, 1]$. It is the response of a classifier. Moreover, a single instance of an object typically generates a large number of positive responses in a neighborhood of this instance (examples are provided in Figure 5.4). The number of positive responses depends on the size, in pixel of these instances. A detailed modeling of this process would require a hierarchical model and several parameters. Instead, we opt for a simplified model in which a high probability corresponds to the situation where the observed value y_m is close to the intensity λ_m. We also want this likelihood function $\ln p(y|g)$ to be concave so that its summation with the prior logarithm $\ln p(g)$ results in a concave function with a unique maximum. Eq. (2.3) below provides one such function:

$$\ln p(y_m|g_m) \propto -\beta(g_m - \sqrt{y_m})^{2p}, \tag{2.3}$$

for some $p \in \{1, 2, \ldots\}$, and $\beta > 0$ Note that the case $p = 1$ corresponds to a Normal distribution. The case $p = 2$ is also interesting. In this case, $p(y_m|g_m)$ has a "plateau" centered at $\sqrt{y_m}$. The size of the plateau depends on β. This is the model that we use in our experiments.

We now assume that for each $1 \leq m \leq \tilde{n}$,

$$\begin{aligned} \ln p(y|g) &= \sum_{m=1}^{\tilde{n}} \ln p(y_m|g) \\ &= \sum_{m=1}^{\tilde{n}} \ln p(y_m|g_m) \end{aligned} \tag{2.4}$$

In other words, the observations are conditionally independent given the intensity, and the observation y_m depends on λ only through λ_m, the intensity at x_m. We also assume that $\ln p(y_m|g_m)$ is a concave function of g_m. This is the case for the traditional Cox process, that is when y_m is Poisson distributed with intensity λ_m and $\phi(g) = g^2$. This is also the case for the model presented in Eq. (2.3).

2.6 The Posterior Distribution

The posterior on g given the observations y is,

$$\ln p(g|y) \propto \ln p(g) + \ln p(y|g)$$

Since in general $p(y|g)$ is non-Gaussian, $p(g|y)$ is non-Gaussian, and thus cannot be computed analytically. Instead, following [39] and [36], we use a sophisticated numerical method which provides a tractable approximation of the distribution of $p(g|y)$. Specifically, we use the Laplace method together

with Kronecker algebra and pre-conditioning to compute a Gaussian approximation of $p(g|y)$. The posterior mean of g given y is notated $\hat{g}$ and the (n, n) posterior covariance matrix of g is notated A.

Let $\Phi(g) = \ln p(g|y)$; in order to find the posterior mean $\hat{g}$ that maximizes the log posterior $\Phi(g)$, Laplace approximation uses the second order Taylor series expansion of $\Phi(g)$ about the point $g = \hat{g}$,

$$\Phi(g) \simeq \Phi(\hat{g}) + \frac{1}{2}(g - \hat{g})^T \nabla\nabla\Phi(\hat{g})(g - \hat{g}) \tag{2.5}$$

Note that the first order term in Eq. (2.5) is zero since the gradient, $\nabla\Phi(\hat{g}) = 0$ at the maximum. Noting that Eq. (2.5) is log-Gaussian, we get,

$$p(g|y) \approx \mathcal{N}(\hat{g}, -(\nabla\nabla\Phi(\hat{g}))^{-1}) = \mathcal{N}(\hat{g}, A) \tag{2.6}$$

Differentiating $\Phi(g)$ w.r.t. g provides

$$\begin{aligned} \nabla\Phi(g) &= -\Sigma^{-1}(g - \bar{g}) + \nabla_g \ln p(y|g) \\ \nabla\nabla\Phi(g) &= -\Sigma^{-1} + \nabla\nabla_g \ln p(y|g) \\ &= -\Sigma^{-1} - W \end{aligned} \tag{2.7}$$

where $W = -\nabla\nabla_g \ln p(y|g)$. The posterior covariance is $A = (\Sigma^{-1} + W)^{-1}$. Note that since $\ln p(y_m|g_m)$ is a concave function of g_m, W is semi-definite positive and A is well defined. (See Section. 2.5 and Appendix A.2)

In the Laplace method, $\hat{g}$ is computed using Newton's algorithm. Following [36], the Newton iteration for the Laplace Approximation can be

computed in the following manner which improves the numerical stability of the algorithm:

$$
\begin{aligned}
g^{(t+1)} &= g^{(t)} - (\nabla\nabla\Phi(g^{(t)}))^{-1}\nabla\Phi(g^{(t)}) \\
&= \left(\Sigma^{-1} + W\right)^{-1}\Big[Wg^{(t)} + \\
&\Sigma^{-1}\bar{g} + \nabla_g \ln p(y|g^{(t)})\Big] \\
&= \Sigma\left(I - W^{1/2}B^{-1}W^{1/2}\Sigma\right)\Big[Wg^{(t)} + \\
&\Sigma^{-1}\bar{g} + \nabla_g \ln p(y|g^{(t)})\Big] \\
&= \Sigma\Big[b - W^{1/2}B^{-1}W^{1/2}\Sigma b\Big] \\
&= \Sigma a
\end{aligned}
\tag{2.8}
$$

where, $B = I + W^{1/2}\Sigma W^{1/2}$, $a = b - W^{1/2}B^{-1}W^{1/2}\Sigma b$, $b = Wg^{(t)} + \Sigma^{-1}\bar{g} + \nabla_g \ln p(y|g^{(t)})$.

Note that W is a non-negative diagonal matrix and B is symmetric positive definite.

Chapter 3

Strategies For Performance Optimization

The Cox formulation described above in Sect. 2.6 is computationally inefficient as it requires $O(n^2)$ storage and $O(n^3)$ computation on two dimensional data. The Newton's step involves inversions and vector product multiplications on covariance matrices. For a (1024×1024) image, the full covariance matrix is of size $(2^{20} \times 2^{20})$, which needs approximately 1 terabyte of memory. Moreover, inversions of such matrices on a CPU-only machine is of the order of two minutes. Since most computer visions systems need much faster response times, we discuss some strategies that can be employed to make the posterior computation more efficient. We show that significant efficiency gains can be obtained as long as the input locations $\mathcal{D}$ lie on a Cartesian grid *i.e.* $\mathcal{D} = \mathcal{D}_1 \times \ldots \times \mathcal{D}_d$. This requirement does not present a limitation for image data as the Cartesian grid structure is natural for images. Since the covariance matrix is the main bottleneck, both in terms of storage and computation, we specifically look at ways to represent and manipulate them using alternate means.

3.1 Separable Kernels

Solving Eq. (2.8) directly is not practical since it requires manipulation of matrices of size (n, n), where n is the number of pixels. To tackle this, we make use of covariance functions that can be decomposed as a product of separable functions over dimensions $i = 1, \ldots, d$ as,

$$k(x, x') = \prod_{i=1}^{d} k_i(x^{(i)}, x'^{(i)}) \tag{3.1}$$

where $x^{(i)}$ is the i^{th} dimensional element of input x.

Such kernels are called tensor product kernels. Note that it is a requirement of the method (see Algorithm 2 and 3) that the covariance kernel be decomposable as a product. While this is not true for all tensor product kernels, the number of compliant kernels is sufficiently large that this does not present a serious limitation. Examples include the exponential kernel, squared exponential kernel and the Matérn kernel (See [36] and [41]). The squared

exponential kernel can be decomposed as:

$$
\begin{aligned}
k(x, x') &= \sigma^2 exp(-\frac{\|x - x'\|^2}{2l^2}) \\
&= \sigma^2 exp\left(-\sum_{i=1}^{d} \frac{\left(x^{(i)} - x'^{(i)}\right)^2}{2l^2}\right) \\
&= \prod_{i=1}^{d} \sigma^{2/d} exp\left(-\frac{\left(x^{(i)} - x'^{(i)}\right)^2}{2l^2}\right) \\
&= \prod_{i=1}^{d} k_i(x^{(i)}, x'^{(i)})
\end{aligned}
\tag{3.2}
$$

The k_i's in Eq. (3.2) are squared exponential kernels over scalar inputs with amplitude $\sigma^{2/d}$. Each k_i only depends on a single dimension but their product gives rise to a prior that spans all d dimensions. We now see how this type of tensor product kernels can be represented in matrix form using Kronecker methods.

3.2 Kronecker Algebra

In this section, we review the basic properties of Kronecker products. Kronecker product, also known as a *direct product* or a *tensor product*, has its origin in group theory and has important applications in fields such as systems theory, signal processing, matrix calculus and particle physics [67, 68]. It is considered to be a generalization of vector outer product to matrices, and is defined as follows:

Definition 1. *Consider a matrix* $A = [a_{ij}]$ *of order* $(m \times n)$ *and a matrix* $B = [b_{ij}]$

of order $(p \times q)$. *The Kronecker product of the two matrices, denoted by* $A \bigotimes B$ *is the* $mp \times nq$ *block matrix,*

$$A \bigotimes B = \begin{bmatrix} a_{11}B & a_{12}B & \dots & a_{1n}B \\ a_{21}B & a_{22}B & \dots & a_{2n}B \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1}B & a_{m2}B & \dots & a_{mn}B \end{bmatrix}$$

$A \bigotimes B$ has mn blocks, the $(i,j)^{th}$ block is the matrix $[a_{ij}]B$ of size $(p \times q)$. For example, let

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

then,

$$A \bigotimes B = \begin{bmatrix} a_{11}B & a_{12}B \\ a_{21}B & a_{22}B \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} & a_{11}b_{12} & a_{12}b_{11} & a_{12}b_{12} \\ a_{11}b_{21} & a_{11}b_{22} & a_{12}b_{21} & a_{12}b_{22} \\ a_{21}b_{11} & a_{21}b_{12} & a_{22}b_{11} & a_{22}b_{12} \\ a_{21}b_{21} & a_{21}b_{22} & a_{22}b_{21} & a_{22}b_{22} \end{bmatrix}$$

3.3 Properties of the Kronecker Product

Table 3.1 lists the basic properties of Kronecker matrix products. We now provide proofs for the relevant properties that we use in our algorithms.

Mixed Product Rule

Property (3.5) $(A \bigotimes B)(C \bigotimes D) = (AC \bigotimes BD)$ is called the mixed product rule because it combines ordinary matrix product and the Kronecker product.

Proof. The $(i,j)^{th}$ block of the product on the left hand side is obtained by taking the product of the i^{th} row block of $(A \bigotimes B)$ and the j^{th} column block of

Table 3.1: Properties of the Kronecker Product

Scalar Product	$A \bigotimes \alpha B = \alpha A \bigotimes B$, where α is scalar	(3.1)
Bilinearity	$A \bigotimes (B + C) = A \bigotimes B + A \bigotimes C$	(3.2)
Associativity	$(A \bigotimes B) \bigotimes C = A \bigotimes (B \bigotimes C)$	(3.3)
Transpose	$(A \bigotimes B)^T = A^T \bigotimes B^T$	(3.4)
Mixed-Product	$(A \bigotimes B)(C \bigotimes D) = (AC \bigotimes BD)$	(3.5)
Inverse	$(A \bigotimes B)^{-1} = A^{-1} \bigotimes B^{-1}$	(3.6)
Determinant	$det(A \bigotimes B) = det(A)^n det(B)^m$, where A is (m,m) and B is (n,n)	(3.7)
Vectorization	$vec(AXB) = (B^T \bigotimes A) vec(X)$	(3.8)
Trace	$trace(A \bigotimes B) = trace(A) trace(B)$	(3.9)

$(C \bigotimes D)$, and this is of the form:

$$\begin{bmatrix} a_{i1}B & a_{i2}B & \dots & a_{in}B \end{bmatrix} \begin{bmatrix} c_{1j}D \\ c_{2j}D \\ \vdots \\ c_{n1}D \end{bmatrix} = \sum_r a_{ir} c_{rj} BD$$

The $(i,j)^{th}$ block of the right hand side is $g_{ij}BD$, where g_{ij} is the $(i,j)^{th}$ element of the matrix AC. By the rule of matrix multiplications,

$$g_{ij} = \sum_r a_{ir} c_{rj}$$

Since the $(i,j)^{th}$ blocks left and right hand sides are equal, the result follows.

□

Vectorization

This property is used to efficiently compute the product Σa in the Newton iteration Eq. (2.8) as shown in Algorithm 1. Σ is the covariance matrix and a is a column vector. We now provide the proof of this property:

$vec(AXB) = (B^T \bigotimes A)vec(X)$.

Proof. Here the operator *vec* refers to the column-wise stacking of the elements of a matrix. Let A, X and B be of order $(p \times q)$, $(q \times m)$ and $(m \times n)$ respectively. Representing the matrices B and X in terms of their column vectors, we get,

$$B = \begin{bmatrix} b_1 & b_2 & \dots & b_n \end{bmatrix} \text{ and } X = \begin{bmatrix} x_1 & x_2 & \dots & x_m \end{bmatrix}$$

Let us denote the k^{th} column of the product AXB as $(AXB)_{:,k}$

$$(AXB)_{:,k} = AXb_k = \sum_{i=1}^{m} x_i b_{i,k}$$

$$= \begin{bmatrix} b_{1k}A & b_{2k}A & \dots & b_{mk}A \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}$$

$$= \begin{bmatrix} b_{1k} & b_{2k} & \dots & b_{mk} \end{bmatrix} \bigotimes Avec(X)$$

Stacking the columns together, we get,

$$vec(AXB) = \begin{bmatrix} AXB_{:,1} \\ AXB_{:,2} \\ \vdots \\ AXB_{:,n} \end{bmatrix} = \begin{bmatrix} b_1^T \bigotimes A \\ b_2^T \bigotimes A \\ \vdots \\ b_n^T \bigotimes A \end{bmatrix} vec(X) = (B^T \bigotimes A) vec(X)$$

This completes the proof. □

Kronecker Inverse

The inverse property (3.6): $(A \bigotimes B)^{-1} = A^{-1} \bigotimes B^{-1}$ is easily proved using the mixed-product rule (3.5).

Proof. Given $A(m \times m)$ and $B(n \times n)$, and assuming A^{-1} and B^{-1} exist,

Using (3.5),

$(A \bigotimes B)(A^{-1} \bigotimes B^{-1}) = AA^{-1} \bigotimes BB^{-1} = I$

□

Determinant

The proof for the determinant property (3.7) relies upon the fact that the determinant of a matrix is equal to the product of its eigenvalues.

3.4 Covariance Matrix as a Kronecker Product

We now show that any covariance function that can be decomposed as a product of axis-aligned kernel functions (as shown in Eq. 3.2) leads to a covariance matrix that can be expressed as a Kronecker product of d smaller

covariance matrices.

For illustration purposes, consider a two dimensional grid of size $n = n_1 \times n_2$. The covariance matrix Σ is then of size $(n \times n)$. Using the kernel function $k(\mathbf{x_i}, \mathbf{x_j})$ (Eq. 3.3), the full covariance matrix can be written as follows. Note that $\mathbf{x_i}$ and $\mathbf{x_j}$ are two dimensional vectors here.

$$\Sigma = \begin{bmatrix} k(\mathbf{x_1}, \mathbf{x_1}) & \dots & k(\mathbf{x_1}, \mathbf{x_n}) \\ \vdots & & \vdots \\ k(\mathbf{x_n}, \mathbf{x_1}) & \dots & k(\mathbf{x_n}, \mathbf{x_n}) \end{bmatrix}$$

Now let us consider the smaller covariance matrices Σ_1 and Σ_2 that correspond to the horizontal and vertical axes respectively. There are n_1 columns and n_2 rows on the grid. Here x_i and y_i are scalars corresponding to the x and y coordinates.

$$\Sigma_1 = \begin{bmatrix} k_1(x_1, x_1) & \dots & k_1(x_1, x_{n_1}) \\ \vdots & & \vdots \\ k_1(x_{n1}, x_1) & \dots & k_1(x_{n_1}, x_{n_1}) \end{bmatrix}, \; \Sigma_2 = \begin{bmatrix} k_2(y_1, y_1) & \dots & k_2(y_1, y_{n_2}) \\ \vdots & & \vdots \\ k_2(y_{n_2}, y_1) & \dots & k_2(y_{n_2}, y_{n_2}) \end{bmatrix}$$

$$\Sigma_1 \bigotimes \Sigma_2 = \begin{bmatrix} k_1(x_1, x_1)\Sigma_2 & \dots & k_1(x_1, x_{n_1})\Sigma_2 \\ \vdots & & \vdots \\ k_1(x_{n1}, x_1)\Sigma_2 & \dots & k_1(x_{n_1}, x_{n_1})\Sigma_2 \end{bmatrix}$$

$$= \begin{bmatrix} k_1(x_1, x_1)k_2(y_1, y_1) & \dots & k_1(x_1, x_{n_1})k_2(y_1, y_{n_2}) \\ \vdots & & \vdots \\ k_1(x_{n1}, x_1)k_2(y_{n_2}, y_1) & \dots & k_1(x_{n_1}, x_{n_1})k_2(y_{n_2}, y_{n_2}) \end{bmatrix}$$

$$= \begin{bmatrix} k(\mathbf{x_1}, \mathbf{x_1}) & \dots & k(\mathbf{x_1}, \mathbf{x_n}) \\ \vdots & & \vdots \\ k(\mathbf{x_n}, \mathbf{x_1}) & \dots & k(\mathbf{x_n}, \mathbf{x_n}) \end{bmatrix} = \Sigma$$

Extending this to d dimensions, we get:

$$\Sigma = \bigotimes_{i=1}^{d} \Sigma_i \tag{3.3}$$

where Σ_i is the covariance matrix on each dimension and represents the correlation between any two observations along that dimension. The d matrices are of sizes $(n_1, n_1) \dots (n_d, n_d)$, where n_i is the size of $\mathcal{D}_i$ and $n = n_1 \times \dots \times n_d$.

Plugging this result into the Newton Iteration, Eq. (2.8) becomes

$$g^{(t+1)} = \left(\bigotimes_{i=1}^{d} \Sigma_i \right) a \tag{3.4}$$

The time complexity for computing this product using standard matrix-vector multiplication is $O(n^2)$, but it is possible to achieve close to linear runtime by utilizing the vectorization property (3.8) of Kronecker algebra. We use the algorithm from Saatçi [41] to compute this product efficiently, which is reproduced below for completeness (Algorithm 1).

In the algorithm below, Σ_i is a $n_i \times n_i$ matrix and $\mathbf{a}$ is a $n \times 1$ vector. Line 5 reshapes the vector $\mathbf{a}$ as a matrix with n_i rows and n/n_i columns. This

Algorithm 1 Kronecker Vector Product

1: **input:** d matrices $\{\Sigma_1, \Sigma_2, \cdots, \Sigma_d\}$; Vector **a** of length n
2: **output:** $\alpha = \left(\bigotimes_{i=1}^{d} \Sigma_i\right)$ **a**
3: **for** $i \leftarrow d$ **to** 1 **do**
4: $\quad n_i \leftarrow size(\Sigma_i)$
5: $\quad A \leftarrow reshape(a, n_i, n/n_i)$
6: $\quad Z \leftarrow (\Sigma_i A)^T$
7: $\quad a \leftarrow vec(Z)$
8: **end for**
9: $\alpha \leftarrow$ **a**
10: **output:** α

algorithm directly makes use of the vectorization property (3.8) in Line 6.

The Newton's iteration in Eq. (2.8) also involves the inversion of the covariance matrix when the prior mean $\bar{g}$ is non-zero. This is useful in applications where the prior distribution of objects is known in advance. Utilizing the Kronecker inversion property (3.6) reduces the computational complexity for this operation from $O(n^3)$ to $O(n^2)$ for two-dimensional data. The details of this analysis are presented in Section 3.6. We now show that the inversion of the full covariance matrix can be simplified as a product of the inverse of smaller covariance matrices.

Theorem 1. *If* $\Sigma = \bigotimes_{i=1}^{d} \Sigma_i$, *then* $\Sigma^{-1} = \bigotimes_{i=1}^{d} \Sigma_i^{-1}$.

Proof.

$$\begin{aligned}
\Sigma^{-1} &= \left(\bigotimes_{i=1}^{d} \Sigma_i\right)^{-1} \\
&= \left(\Sigma_1 \otimes \bigotimes_{i=2}^{d} \Sigma_i\right)^{-1} \\
&= \Sigma_1^{-1} \otimes \left(\bigotimes_{i=2}^{d} \Sigma_i\right)^{-1} \text{ Using Property (3.6)} \\
&= \bigotimes_{i=1}^{d} \Sigma_i^{-1}
\end{aligned}$$

□

3.5 Algorithms for Posterior Computation

We now incorporate the ideas presented in section 3.4 into the Newton's method, for computational and storage efficiency. In Newton iteration we need to evaluate B^{-1}. The Cholesky decomposition would require $\mathcal{O}(n^3)$ time and $\mathcal{O}(n^2)$ storage. However, due to the Kronecker structure of the covariance matrix Σ, the Conjugate Gradient Method only requires $\mathcal{O}(dn^{\frac{d+1}{d}})$ time and $\mathcal{O}(dn^{\frac{2}{d}})$ storage, which is also employed in Flaxman et al. [39]. We now summarize the Conjugate Gradient Algorithm 2 adapted for the Kronecker structure.

Based on Algorithms 1 and 2, the Newton Iteration in Eq. (2.8) can be implemented by the Algorithm 3. Computing $a = \left[b - W^{1/2}B^{-1}W^{1/2}\Sigma b\right]$ in Eq. (2.8) involves finding B^{-1} which is computationally expensive. To overcome

Algorithm 2 Kronecker Conjugate Gradient

input: $\Sigma_1, \Sigma_2, \cdots, \Sigma_d$, b, W
$h \leftarrow W^{1/2}\left((\bigotimes_{i=1}^{d}\Sigma_i)b\right)$
$x0 \leftarrow \vec{0}$
$r0 \leftarrow h - x0 - W^{1/2}\left((\bigotimes_{i=1}^{d}\Sigma_i)(W^{1/2}x0)\right)$
$d0 \leftarrow r0$
repeat
  $\alpha \leftarrow \frac{r0^T r0}{\left(d0^T d0 + d0^T W^{1/2}\left((\bigotimes_{i=1}^{d}\Sigma_i)(W^{1/2}d0)\right)\right)}$
  $x1 \leftarrow x0 + \alpha d0$
  $r1 \leftarrow r0 - \alpha d0 - \alpha W^{1/2}\left((\bigotimes_{i=1}^{d}\Sigma_i)(W^{1/2}d0)\right)$
  $\beta \leftarrow (r1^T r1)\ /(r0^T r0)$
  $d1 \leftarrow r1 + \beta d0$
  $r0 \leftarrow r1$
  $d0 \leftarrow d1$
  $x0 \leftarrow x1$
until $r0$ is sufficiently close to 0
output: $x1$

this, we set $Bx = W^{1/2}\Sigma b$, and solve for x using Algorithm 2, as shown below in line 6 of Algorithm 3.

Algorithm 3 Kronecker Newton

input: $\Sigma_1, \Sigma_2, \cdots, \Sigma_d$, $y_{1:M}$, $lnp(y_{1:M}|g)$
$g \leftarrow \overrightarrow{0.5}$
repeat
  $W \leftarrow -\nabla\nabla lnp(y_{1:M}|g)$
  $b \leftarrow Wg + \left(\bigotimes_{i=1}^{d}\Sigma_i^{-1}\right)\bar{g} + \nabla_g \ln p(y_{1:M}|g)$
  Solve $Bx = W^{1/2}(\bigotimes_{i=1}^{d}\Sigma_i)b$ with Algorithm 2
  $a \leftarrow b - W^{1/2}x$
  $g \leftarrow \left(\bigotimes_{i=1}^{d}\Sigma_i\right)a$
until convergence

3.6 Complexity Analysis

The size of the covariance matrix determines the upper limit of the storage requirement. If n is the total number of points on the observation grid, the covariance matrix would be of size (n, n), thus making the storage $O(n^2)$. However, as described in section 3.4, if we use a separable kernel and write the covariance matrix Σ as a Kronecker product of d smaller covariance matrices, each of size $(n^{\frac{1}{d}}, n^{\frac{1}{d}})$, the storage now becomes $O(dn^{\frac{2}{d}})$. In the case of a 2-dimensional grid, the storage is O(2n) $\simeq$ O(n). Note that we never explicitly store Σ anywhere in our algorithm, but instead work only with the smaller Σ_i matrices as illustrated in algorithms 1, 3 and 2.

The time complexity on the other hand is dependent upon the limitations imposed by Newton's algorithm Eq. (2.8). The method requires the inversion of the matrix B, which is of size (n, n). Standard approaches like Cholesky decomposition takes $O(n^3)$ time (See [36]).

Now, let us consider Algorithm 3. Line 8 computes the product $\left(\bigotimes_{i=1}^{d} \Sigma)\right) a$ using Algorithm 1. The $(n, 1)$ matrix a is first reshaped to a matrix of size $(n^{\frac{1}{d}}, n^{\frac{d-1}{d}})$ and it is then multiplied with a matrix of size $(n^{\frac{1}{d}}, n^{\frac{1}{d}})$. This multiplication is O($n^{\frac{1}{d}} n^{\frac{1}{d}} n^{\frac{d-1}{d}}$) = O($n^{\frac{d+1}{d}}$). Since there are d iterations in Algorithm 1, the total time for calculating $(\bigotimes_{i=1}^{d} \Sigma)a$ is $O(dn^{\frac{d+1}{d}})$. The computational complexity of the Conjugate Gradient algorithm 2 used in Line 6 of Alg. 3 is dominated by similar matrix-vector multiplications, hence this requires $O(dn^{\frac{d+1}{d}})$ operations as well. The number of iterations for the convergence of Newton's method is typically much less than n, and as a result the overall computation time for our algorithm is $O(dn^{\frac{d+1}{d}})$. In a two-dimensional space,

this value is close to $O(n^{\frac{3}{2}})$.

Chapter 4

Hyperparameter Learning: Method of Moments

Since Cox processes are non-parametric, it has no notion of fitting the data, and *learning* in Cox-Gaussian process is the problem of choosing the functional form for the covariance kernel as well as the values for any hyperparameters [36]. The covariance function is the crucial ingredient in a Cox process model, as it encodes our assumptions about the function which we wish to learn. In supervised learning, the notion of *similarity* between data points is crucial; there is a basic assumption that data points which are close are likely to have similar target values. On the other hand, under the Gaussian process perspective, it is the covariance function that defines nearness or *similarity*. The choice of the covariance function influences the prior assumptions, which in turn affects the posterior intensity and the final estimated count. Hence, it is important to both choose an appropriate kernel function as well as set its hyperparameters in such a manner that reflects our observations on the training dataset. Even within the same class of covariance kernels, the appearance of the kernel can vary drastically with the values of the hyperparameters, as

indicated by Fig. 4.1. Thus setting appropriate values to these parameters is important as it has a significant impact on the performance of the algorithm.

4.1 Overview of Standard Approaches

4.1.1 Maximum Likelihood Estimation

The traditional method for estimating the parameters of the kernel involves using the method of maximum likelihood, see [36]. Here the goal is to find the parameter values that have the highest chance of producing the observed data. However, this method is often computationally expensive and requires a sophisticated numerical implementation.

4.1.2 Grid Search

Grid search is an exhaustive sweep over a manually chosen subset of the hyper-parameter space. Grid search is typically used in conjunction with cross-validation on the training set. The basic idea in cross-validation [36] is to split the training data into two sets, one is used for training and the other, the validation set, is used to select the model and its hyper-parameters. In cases where the training data-set is small, k-fold cross-validation is used: the data is split into k distinct sets, the training is done on $k-1$ sets and the validation is done on the last remaining set; and this process is repeated k times.

Since grid search is a time consuming operation, we developed a strategy for analytically estimating the model parameters using the training data, which we elaborate in the sections below.

4.2 Kernel Parameter Estimation

We use the *method of moments* to estimate the parameters of the kernel. In this this section, we present the details of our derivation. In statistics, *method of moments* is a way of estimating the model parameters by equating the theoretical expression for the moments with the corresponding values observed from the sample data. Our goal here is to find simple analytical expressions for the kernel parameters so as to avoid complex optimizations. We use the squared exponential kernel and the square link function in our experiments. In this section, we derive closed-form equations for the parameters of this kernel.

We notate the squared exponential kernel,

$$k_{SE}(x, x') = \sigma^2 \exp\left(-\frac{\|x - x'\|^2}{2l^2}\right) \tag{4.1}$$

This kernel has two parameters, l and σ. The parameter l of the kernel defines the characteristic length scale (see Rasmussen & Williams [36]). It describes how far one needs to move along a particular axis in input space for the function values to become uncorrelated. It is a measure of *smoothness* of the function: the larger the value of l, the smoother the function appears. σ is the signal standard deviation, which is a scaling factor that roughly determines the distance of the function away from the mean. Figure 4.1 shows how the general appearance the functions sampled from a Gaussian process with a squared exponential covariance function change with the parameters of the kernel.

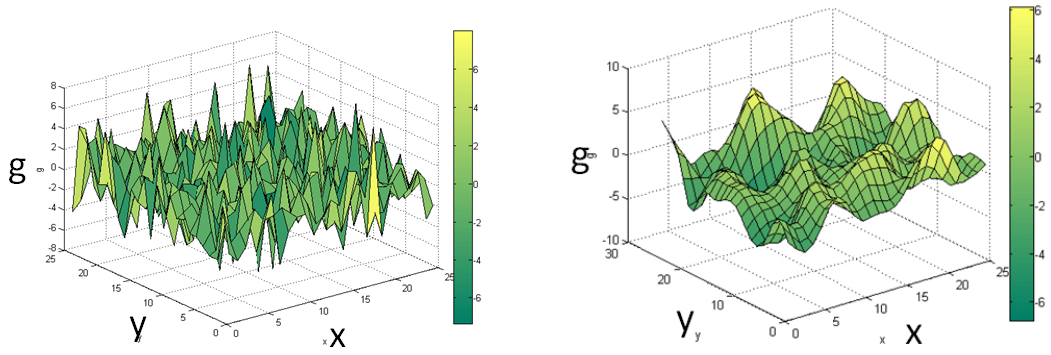


Figure 4.1: Functions with two dimensional input drawn at random from a Gaussian process with squared exponential covariance function, using two different values of the characteristic length scale l. The parameters were: (a) $l = (0.2, 0.2)$, (b) $l = (2, 2)$.

The expression for the k^{th} theoretical moment of the distribution of a random variable X is,

$$\begin{aligned} M_k &= E[X^k], \text{ the } k^{th} \text{ moment about the origin} \\ M_k &= E[(X-\mu)^k], \text{ the } k^{th} \text{ moment about the mean } \mu \\ &\text{where } k = 1, 2 \ldots \end{aligned} \tag{4.2}$$

The *mean* and *variance* are the first two statistical moments. For the second and higher moments, the moment about the mean is typically used rather than the moment about the origin. The third moment is called the *skewness*, which indicates how symmetric the shape of the distribution is. If this value is zero, it means that the distribution is symmetric about its mean. A normal distribution, for instance has a skewness of zero. Roughly speaking, if the distribution has a positive skew, the probability density function has a long tail to the right, such that more of the probability mass is shifted to the left

of the domain, and the curve would appear to be skewed to the left. The opposite behavior is observed when the skew is negative. The fourth moment, known as the *kurtosis* is a measure of the heaviness of the tail of a distribution. The higher moments are significant too, providing valuable information on the shape of the distribution. However, we specifically use only the first and second moments in our estimation. The derivation follows below.

Suppose the image domain is Ω with size $|\Omega| = a \times b$, where a and b are respectively the number of rows and columns in the image. Let $N(\Omega)$ be the number of instances of the object within Ω. According to our Gaussian-Cox model, the number of instances conditional on the intensity, λ is,

$N(\Omega)|\lambda \sim \text{Poisson}\left(\int_\Omega \lambda(s)\, ds\right)$,

where $\lambda(s) = \alpha g^2(s)$ and g is a Gaussian Process such that $g \sim GP\,(0, K)$ and K is a covariance matrix populated using the squared exponential kernel k_{SE} (Eq. 4.1). Note that the scaling factor α used with the link function is the third parameter that needs to be estimated, in addition to σ and l.

Assume that we have m samples $N_1\,(\Omega) \ldots N_m\,(\Omega)$ over the same domain Ω, or over domains of the same size $|\Omega|$. We can then use these samples together with the method of moments to estimate the parameters of the prior α, σ and l as follows. The theoretical mean and variance of $N(\Omega)$ can be notated as $E[N\,(\Omega)]$ and $V[N\,(\Omega)]$. The sample mean $\bar{N}$ and sample variance

S are,

$$\begin{aligned} \bar{N} &= \frac{1}{m}\sum_{i=1}^{m} N_i\left(\Omega\right) \\ S &= \frac{1}{m-1}\sum_{i=1}^{m}\left(N_i\left(\Omega\right)-\bar{N}\right)^2 \end{aligned} \tag{4.3}$$

Note that we need to compute $E[N\left(\Omega\right)]$ and $V[N\left(\Omega\right)]$ as function of the parameters of the model and solve,

$$\begin{aligned} E[N\left(\Omega\right)] &= \bar{N} \\ V[N\left(\Omega\right)] &= S \end{aligned} \tag{4.4}$$

As detailed in the appendix A.1, $E[N\left(\Omega\right)]$ can be easily simplified using the properties of Poisson Processes,

$$\begin{aligned} E\left[N\left(\Omega\right)\right] &= E\left[E\left[N\left(\Omega\right)|\lambda\right]\right] \\ &= \alpha\sigma^2\left|\Omega\right| \end{aligned} \tag{4.5}$$

By the law of total variance, $V[N\left(\Omega\right)]$ can be written as,

$$V\left[N\left(\Omega\right)\right] = V\left[E\left[N\left(\Omega\right)|\lambda\right]\right] + E\left[V\left[N\left(\Omega\right)|\lambda\right]\right] \tag{4.6}$$

It is shown in the appendix A.1 that $V[N\left(\Omega\right)]$ in turn simplifies to,

$$V\left[N\left(\Omega\right)\right] = \alpha^2\sigma^4 2\pi\left|\Omega\right| l^2 + \alpha\sigma^2\left|\Omega\right| \tag{4.7}$$

Setting $\alpha = 1$, and using Eq. 4.3, 4.4, 4.5 and 4.7 , we estimate σ and l as,

$$\hat{\sigma} = \sqrt{\frac{\bar{N}}{|\Omega|}}$$
$$\hat{l} = \sqrt{\frac{|\Omega|\,(S - \bar{N})}{2\pi\bar{N}^2}} \tag{4.8}$$

4.3 Validation

We now verify the method of moments (MoM) using simulated data and real images from MS COCO [24] dataset. For the simulation, we set a range of values for the parameters σ and l of the squared exponential kernel, and sample object instances using Cox processes in 1000 images of size $N \times N$ according to this prior. Specifically, we draw 1000 samples g from multivariate normal distributions characterized by the kernel, and calculate the intensity λ from g using the link function $\phi(g) = g^2$. The scale factor α is set to 1. Note that both g and λ are of the same size as the image. Object instances are then assigned to each image based on this intensity λ; *i.e.* we draw N^2 samples from a uniform distribution and assign an instance wherever the intensity exceeds the sample value. The assumption here is that the locations with higher intensity values will have higher chances to have an object. The number of instances in each image is N_i, from which we compute the sample mean $\bar{N}$ and sample variance S. We then use Eq. 4.8 to find $\hat{l}$ and $\hat{\sigma}$. We repeat this process 100 times, estimating $\hat{l}$ and $\hat{\sigma}$ in every iteration of 1000 images each. Table 4.1 shows the mean and standard deviation of the estimates $\hat{l}$ and $\hat{\sigma}$ over 100 iterations. In the first column of the table, we fix the value for $l = 1.5$ and use different values for σ. The second column shows the mean and

standard deviation of the estimates of $\hat{\sigma}$ corresponding to each *true* σ value in the first column. Column 3 and 4 show similar results for the parameter l. These results indicate that the method of moments provide good estimates for the kernel parameters. The histogram of the estimates of $(\hat{\sigma}, \hat{l})$ over 100 iterations are shown in Figure 4.2.

Table 4.1: MoM estimates for prior parameters σ and l

$l = 1.5$	$\hat{\sigma}$ mean(std)	$\sigma = 0.1$	$\hat{l}$ mean(std)
$\sigma = 0.1$	0.09 (0.0005)	$l = 0.5$	0.81 (0.39)
$\sigma = 0.5$	0.50 (0.0011)	$l = 1.0$	0.97 (0.36)
$\sigma = 1.0$	1.00 (0.0023)	$l = 1.5$	1.41 (0.29)
$\sigma = 1.5$	1.49 (0.0030)	$l = 2.0$	1.92 (0.19)
$\sigma = 2.0$	1.99 (0.0038)	$l = 3.0$	2.81 (0.19)

Next, we performed experiments on real images from MS-COCO "bird" dataset. First, the mean and variance of the number of birds per image were determined on the training dataset using ground truth annotations. From the sample mean and variance, σ and l were estimated using Eq. 4.8. Since we do not have ground truth kernel parameter values for real images, we use the final results from the counting algorithm to verify our results. We used a large range of arbitrary parameter values, including the estimates from MoM, and computed the estimated number of birds in each image using the algorithms presented in chapter 3, section 3.5. Our results confirmed that the best counting performances were obtained when the parameter values were close to the ones estimated via MoM.

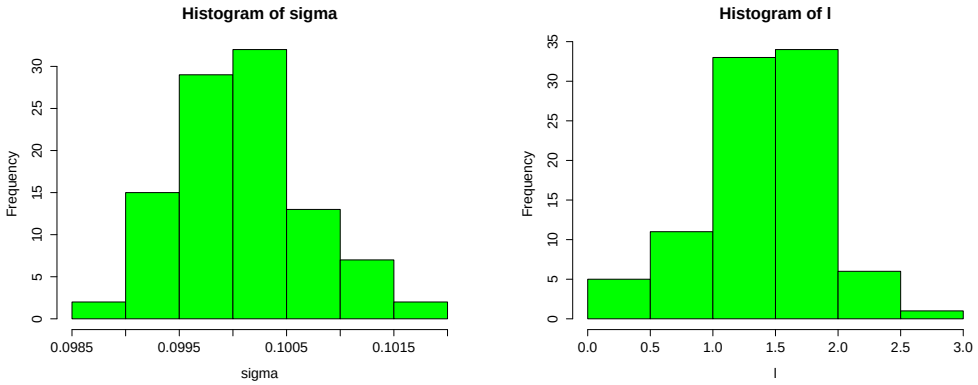


Figure 4.2: (left) Histogram of estimated $\hat{\sigma}$ on 100 simulation samples. True value of $\sigma = 0.1$ (right) Similar histogram for the estimates of the parameter $\hat{l}$. True values of $l = 1$.

Chapter 5

Counting Experiments

5.1 Simulation

In this section, we demonstrate our algorithm on simulated image data.

In order to simulate the computation of the posterior intensity and the subsequent estimation of the number of object instances, we generate images of size (30×30) over which instances are assigned using the following scheme:

- Sample according to a Poisson process over the image domain with constant intensity $\lambda = 5/(30 \times 30)$ such that on average, 5 instances per image are sampled.
- Set a bounding box centered at each object instance with a random width and height ranging from 3 to 10.
- In real-world images, the positive responses generated by the CNN classifier are clustered in blob-like structures as shown in the second column of Figure 5.4. To simulate this behavior, we generate object instances over the bounding box using a Bernoulli distribution with probability p.

Specifically, since each instance generates multiple measurements, and in order that the total intensity be close to the expected number of instances, we set each measurement y_m in the bounding box to $y_m = 1/(p * N)$ with probability $p = 0.8$ and $y_m = 0$ with probability $1 - p = 0.2$ (where N is the total number of measurements in the bounding box.)

We use the squared exponential kernel, $k_{SE}(x, x') = \sigma^2 \exp\left(-\frac{\|x-x'\|^2}{2l^2}\right)$ with parameters $\sigma = 2.4$ and $l = 2$ as the GP covariance function for this simulation. The top image of Figure 5.1 shows a function drawn at random from the GP prior, and the image below is the corresponding positive prior intensity.

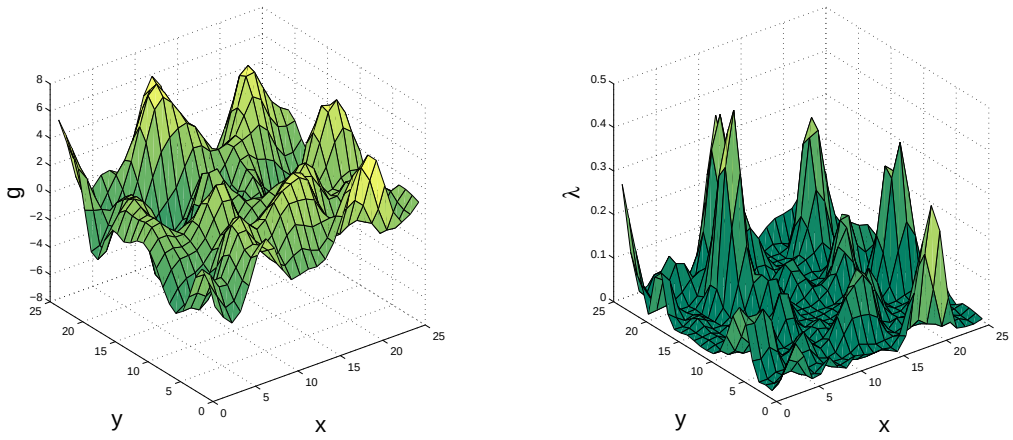


Figure 5.1: (left) The prior function g drawn at random from a GP with a squared exponential covariance function; the input here is two dimensional, and (right) the corresponding positive prior intensity, $\lambda = \alpha g^2$

Fig. 5.2 shows the results of one of the simulations. The red triangles in each bounding box stand for the detected instances of the object; this simulates the classifier results as there are typically multiple hits per instance. There are some missed measurements in each bounding box as these are populated

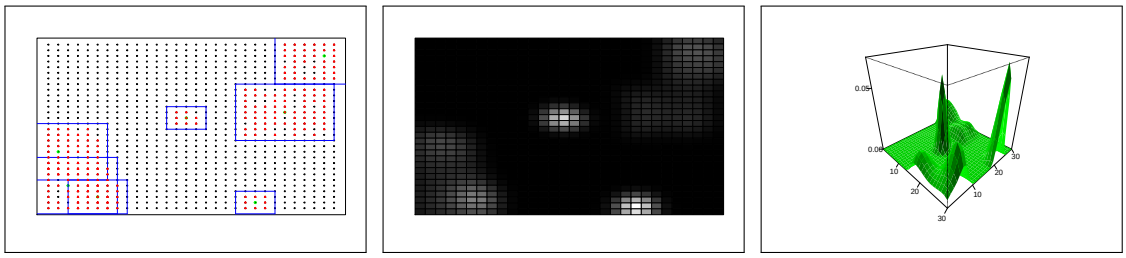


Figure 5.2: Simulation results: (left) A simulated input image with 8 object instances, some of them overlapping. The blue rectangles show the bounding box annotations around the object. The red triangles in each bounding box stand for the detected instances of the object; this simulates the classifier results as there are typically multiple hits per instance. (middle) The posterior intensity computed by our algorithm and (right) The same posterior intensity as a 3D plot.

randomly by a Bernoulli distribution. This is done in order to simulate the behavior that the classifier would return a true detection only for a fraction of the measurements within the bounding box. The middle image in Figure 5.2 shows the posterior intensity - the black pixels indicate that the corresponding intensity values at those locations are close to 0. The rightmost image of Figure 5.2 displays the posterior intensity in a 3D graph. Note that Figure 5.2 demonstrates the situation where some of the object instances partially overlap and this overlapping is also reflected clearly in the posterior intensity in the same figure.

We replicate this procedure two thousand times and compute the total posterior intensity for each iteration. A standard linear regression equation is then used to model the relation between the total intensity and the true number of instances. Note that a more sophisticated forward model than the one described in Sect. 2.5 would potentially allow to compute the number of instances simply by integrating the estimated intensity. However, this would

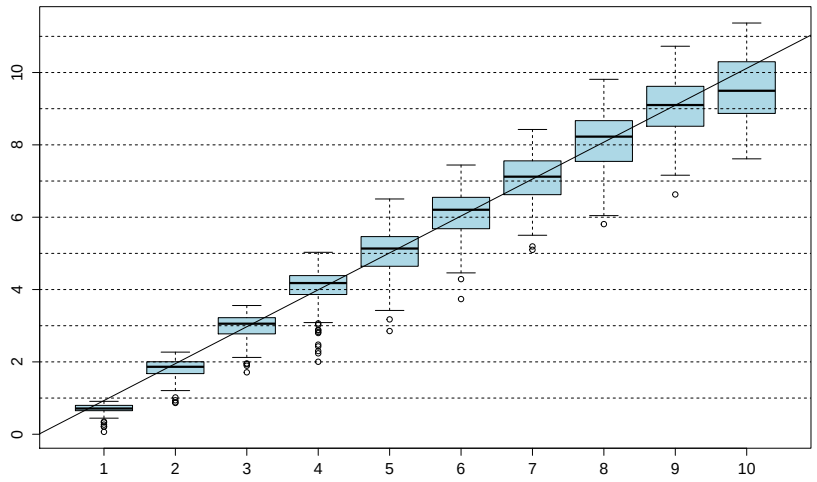


Figure 5.3: Counting results on simulated data: The true count(x-axis) is plotted against the count estimated(y-axis) by the proposed algorithm over 2000 simulated images. (The number of object instances per image varies from 1 to 10 in this simulation.)

likely be obtained at some non negligible computation cost. According to the regression equation, we estimate the number of objects in each simulated image. The results are presented in Figure 5.3, the horizontal axis indicates the actual number of instances and the vertical axis is the estimated count. This box plot shows that the median of the estimated count is very close to the actual number of instances.

5.2 Application to Real Images

We now present the results of our algorithm on images from the MS COCO [24] dataset. In general, our approach works for all categories of objects, provided we have a classifier for that particular class. In this section, we

demonstrate our results on three classes of objects from MS COCO dataset: “bird”, “motorcycle” and “sheep”. These classes were specifically chosen due to their datasets containing a satisfactory number of images with multiple instances of the same object type. In contrast, the “cat" or “table" datasets are predominantly comprised of solitary-instance images, and hence are not particularly challenging for our algorithm. (See the first column of Figure 5.4 for example images from COCO dataset.)

We use the squared exponential kernel as the covariance function for the GP prior, estimating the parameters σ and l using Eq. (6.2) derived in section 4.2. The sample mean and standard deviation used in this equation are computed using the ground truth bounding boxes annotations from COCO dataset. This is done for each class separately.

We begin the classification process by dividing the input image of size $M \times N$ into a $\frac{M}{\Delta t_h} \times \frac{N}{\Delta t_w} = 100 \times 100$ grid, where Δt_w and Δt_h are the pixel distances between grid centers along the width and height of the image respectively. We use a 100×100 grid for all of our experiments with MS COCO. The grid dimension is $d = 2$, the size of the grid is $n = n_1 \times n_2 = 10,000$, and Σ_i is a matrix of size 100×100. Note that the full covariance matrix Σ would be of size $10,000 \times 10,000$, but we neither store nor manipulate this large Σ directly, as explained in Sect. 6. Each point in the grid is the center of an observation bounding box.

We then observe each grid center m at 5 different bounding box sizes by running a classifier trained specifically for the object category of interest. If

none of the bounding boxes contain the object instance, we set the corresponding answer y_m to 0. In real-world images, the positive responses generated by the CNN classifier are clustered in blob-like structures as shown in the second column of Figure 5.4. Since each instance generates multiple measurements, and in order that the total intensity be close to the expected number of instances, we scale the classification score to get the observations y_m. If there is an instance detected at grid location m, we set $y_m = \frac{\Delta t_h \times \Delta t_w \times s}{M_m \times N_m \times p}$, where $\Delta t_h = \frac{M}{100}$ and $\Delta t_w = \frac{N}{100}$ are the grid distances, s is the classification score, $M_m \times N_m$ is the size of the bounding box with the maximum classifier score and p is the percentage of boxes in the neighborhood for which the classifier returns a true detection.

After we collect all the answers y, we compute the posterior distribution using Algorithms 1, 2, and 3 from chapter 3, section 3.5. The average run time of our posterior computation algorithm implemented in MATLAB is 0.04 seconds on a CPU-only Intel Xeon desktop with 8 GB RAM. This was measured on MS COCO dataset that has a typical image size of 500×500 pixels.

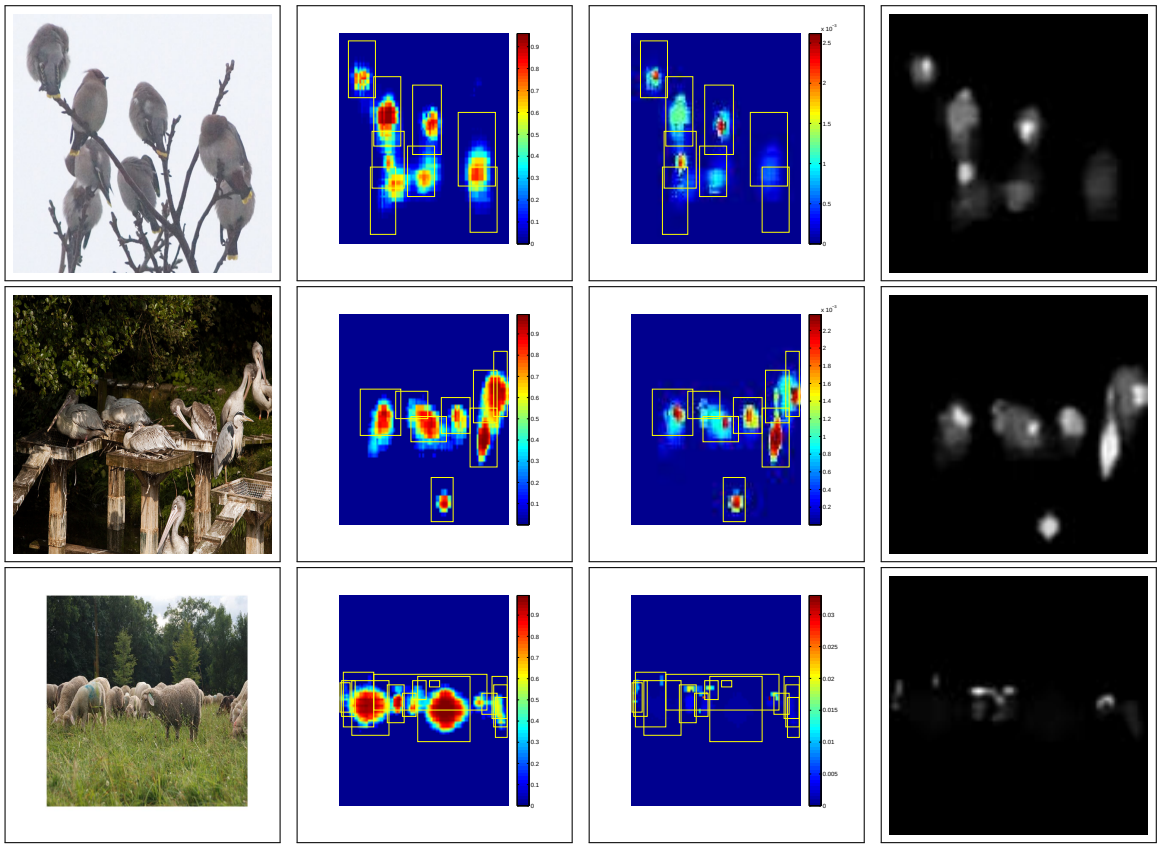
Figure 5.4 shows two sample images from the test dataset, their ground truth bounding boxes in yellow, the answers y from the classifier, and the posterior intensities computed by our algorithm. In order to estimate the final count, we calculate the integral of the posterior intensity over the space of bounding boxes, which is then fitted using linear regression. The results are shown in Figure 5.5 and Table 5.1. The box plots in Figure 5.5 show the true counts plotted against the estimated counts computed by both Cox

and Faster RCNN on the MS COCO *bird*, *sheep* and *motorcycle* datasets. Our algorithm is robust to overlap, crowding and occlusion, the typical scenarios in which *counting by detection* fails. The last row of Figure 5.4 shows a difficult overlapping case for class type: "sheep". As is evident from the last column of Figure 5.4, our method can perform soft localization as well if we post-process the posterior intensity, for instance, by performing a non-maximal suppression over the set of bounding boxes. We intend to explore this framework for full localization in future work.

Table 5.1 details the *root mean square error*(RMSE) between the true number of objects and the count estimated by *Cox*, *Faster-RCNN* and *Faster-RCNN with Regression*. For the last baseline, we perform a linear regression on the final results of *Faster-RCNN*. These numbers are shown in the same table, in the *"Faster RCNN Reg"* row. For each category, we compute the RMSE on bootstrap samples and report the mean and standard deviation over all bootstrap iterations. The number of object instances per image, $k = [1, 14]$; for each bootstrap sample we also compute RMSE separately for each k, find the mean per sample and finally calculate the mean and variance over all iterations. The result of these computations is shown in the last three rows of Table 5.1. The lower error rate observed for Cox over Faster RCNN in all cases suggests that our proposed counting algorithm is promising.

Table 5.1: RMSE mean (std) for counting.
(The lowest error in each category shown in bold font.)

Method	Bird	Motorbike	Sheep
Cox	**1.85** (0.07)	**1.13** (0.14)	**1.44** (0.07)
Faster RCNN	2.00 (0.25)	1.37 (0.20)	2.06 (0.16)
Faster RCNN Reg	2.01 (0.17)	1.15 (0.14)	1.62 (0.11)
Cox k	**2.33** (0.12)	**1.86** (0.34)	**1.84** (0.12)
Faster RCNN k	2.77 (0.29)	2.79 (0.42)	3.03(0.32)
Faster RCNN Reg k	2.53 (0.13)	1.99 (0.38)	2.04(0.22)



(a) (b) (c) (d)

Figure 5.4: Computation of the posterior intensity: (a) 3 sample images from the MS COCO dataset belonging to "bird" and "sheep" class; (b) Classification results: the classification scores after running the CNN classifier on a 100×100 grid, red and blue values indicating high and low confidence respectively; the ground truth bounding boxes annotations are shown in yellow, (c) the scaled classification results y and (d) their corresponding posterior mean intensity computed by our algorithm.

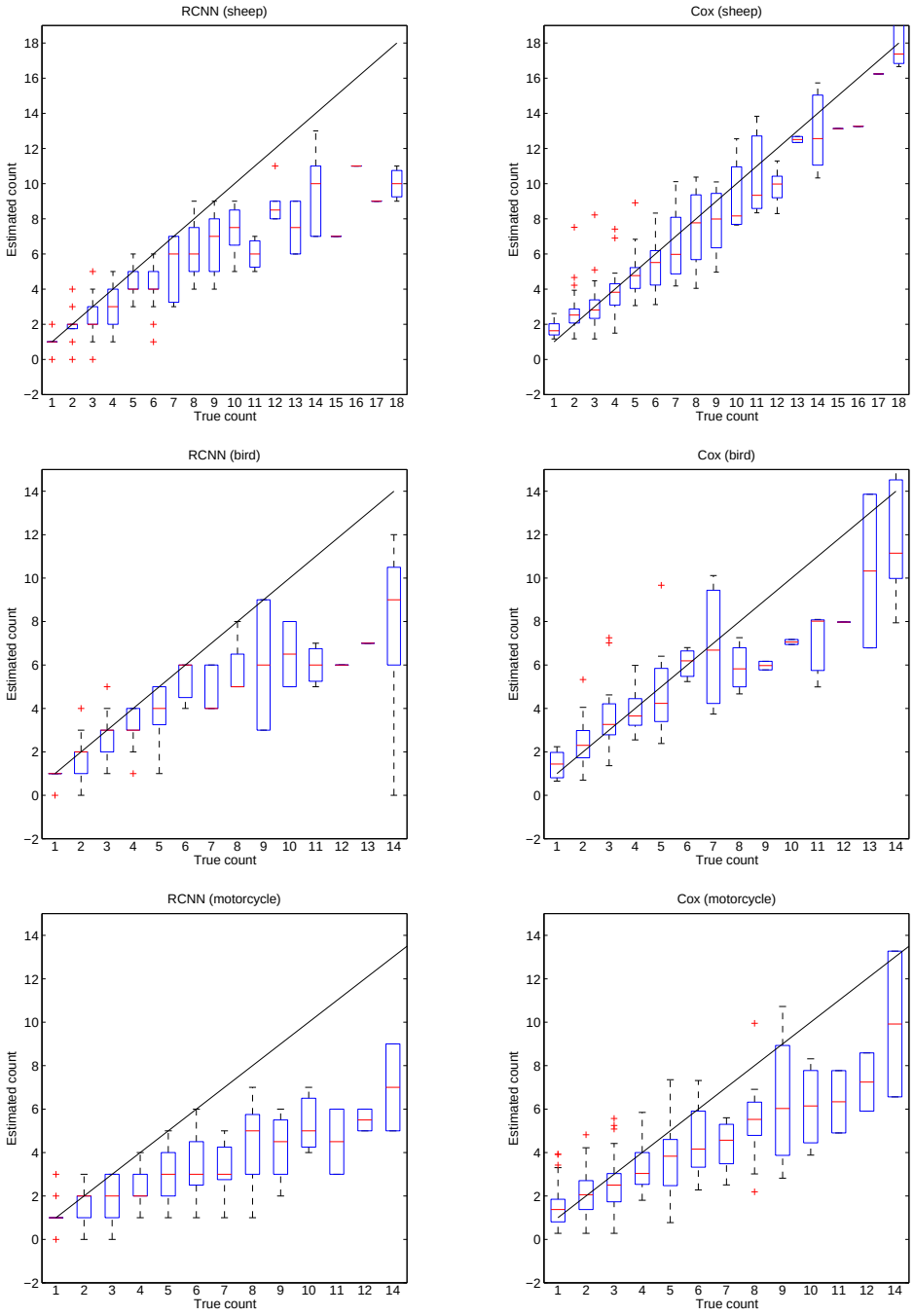


Figure 5.5: Box plots that compare the counts estimated by *Cox* and *Faster RCNN* on images from MS COCO dataset. The number of instances per image ranges from 1 to 14 for both "bird" and "motorcycle" class, while it goes up to 18 for "sheep". On each box, the central red horizontal mark indicates the median, and the bottom and top edges of the box indicate the 25th and 75th percentiles, respectively.
(left) *Faster RCNN* count results. (right) *Cox* results on the same dataset.

Chapter 6

Enhanced Region Proposal Networks for Object Counting

Most object detection and classification systems rely upon the use of region proposal networks or upon classifying the *objectness* of specific sub-windows to help detect potential object locations within an image. In this chapter, we outline a framework that converts such region proposals to a well defined Poisson intensity using the algorithms outlined in chapter 2. This output can be used as-is to directly estimate object counts, or can be *plugged* into pre-existing object detection frameworks to improve their counting and detection performance. This remapping does not require the original network to be re-trained: the parameters of the model can be estimated analytically from the training data. The Cox (doubly stochastic Poisson) process model takes as input a set of bounding boxes and scores, and computes a posterior distribution over the intensity of a Cox process, which can then be used to estimate object counts. In this chapter, we combine the ideas presented in chapter 2 with region proposals from networks such as "Faster RCNN" for improved object counting: *i.e.* instead of selecting the bounding box centers

on a grid, we use proposals from a pre-trained detection network for our experiments. This mapping of region proposals to Poisson intensity can be employed on any counting or detection network that uses *region proposals* in one of the intermediate layers.

6.1 Overview of the Counting Framework

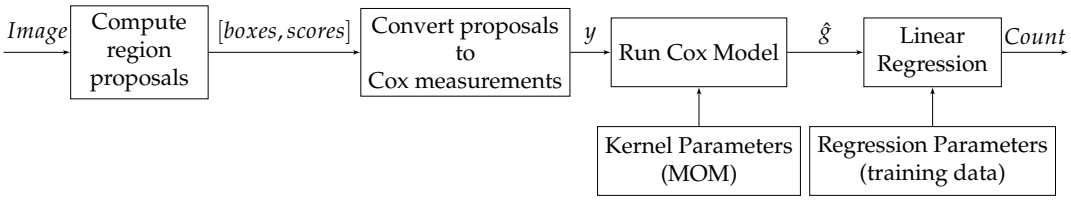


Figure 6.1: Block diagram for the counting work-flow using region proposals.

The block diagram in Fig. 6.1 provides an overview of our counting system. First, we compute an initial set of bounding boxes and *objectness* scores from the input image. These could be region proposals from a detection network such as "Faster RCNN"; or the results from running a trained bounding box object classifier on any arbitrary set of boxes chosen over the image. These proposals are then converted to a Poisson intensity using the model proposed in chapter 2 [30]. In this doubly stochastic Poisson (or Cox) framework, these proposal scores y are modeled to be functionally related to the intensity of the Poisson process. Using the proposals scores y, we then compute a posterior distribution over the intensity of a Cox process. The sum of this posterior intensity would give an estimate of the expected number of objects in that

image.

6.2 Cox Input Scores from Region Proposals

The Poisson model described in chapter 2 uses measurements y as input. There is no restriction on the measurement type, loosely speaking, y_i is a measure of *objectness* at image location i. These initial measurements can be computed as either from region proposals or from running an object classifier over a grid.

To illustrate this idea, we use the region proposal scores from "Faster RCNN". These are in the form of $[boxes, scores]$ for each of the 20 Pascal VOC [22] classes. These proposals are first converted to measurements for the Cox model by employing the following strategy: for each location in the image, we first find the subset of bounding boxes that include the location, and then pick up the maximum score for the object class of interest. An example of this result is shown in the second column of Fig. 6.3. We then convert these measurements to a Poisson intensity by following Algorithms 1, 2 and 3 in Sects. 3.4 and 3.5. This intensity is shown in the last column of Fig. 6.3 for two object categories. After this, we integrate the intensity and re-scale this in order to find an estimate for the expected number of objects in the image. We repeat the process for each of the 20 classes.

Note that we can also convert the Cox intensity back to region proposal scores and continue the "Faster RCNN" flow as-is for improved detection and counting. As per the definition of Poisson processes, the sum of the posterior

intensity over a sub-image provides the expected value of the number of instances in this sub-region. We use this to update the RPN scores, by adding up the posterior mean intensity over each of the bounding box in the region proposal. This can be done efficiently using integral images. This strategy can improve the detection and counting performance of any network that uses region proposals in one of its intermediate layers, as our algorithm converts these proposals to a well-defined Poisson intensity.

6.3 Counting Experiments

We validate our algorithms on the MS COCO [24] dataset. The validation set contains 40,504 images and 80 categories of objects. Of these, we use 20 categories cited in Table 6.2 for our experiments. "Faster RCNN" [21] is the baseline for our object count performance.

Method	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
FRCNN	**0.94**	2.40	4.03	2.67	2.63	1.33	3.01	0.59	3.52	4.09	1.15	0.78	1.84	2.36	3.01	1.69	5.20	1.09	0.70	1.38
Cox	0.98	**1.89**	**3.53**	**2.62**	**2.30**	**1.20**	**2.61**	**0.35**	**0.96**	**3.10**	**0.75**	**0.69**	**1.66**	**2.31**	**2.77**	**1.54**	**4.61**	**0.52**	**0.54**	**1.00**

Table 6.1: RMSE bootstrap mean (std) for 20 object categories. The inputs y for Cox are the scores from a FRCNN's region proposals. The lowest errors in each category are highlighted.

Method	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
FRCNN	**1.32**	3.17	4.56	3.01	2.98	1.84	3.34	0.76	4.06	4.02	1.73	0.99	2.23	2.93	2.95	2.09	4.31	1.32	0.89	1.62
Cox	**1.41**	**2.27**	**3.79**	**2.84**	**2.34**	**1.64**	**2.61**	**0.59**	**3.00**	**2.98**	**1.15**	**0.96**	**1.92**	**2.63**	**2.66**	**1.86**	**3.96**	**0.70**	**0.84**	**1.46**

Table 6.2: RMSE-k bootstrap mean (std) for 20 object categories; computed separately for each value of object count k and then averaged. The inputs y for Cox are the scores from a FRCNN's region proposals. The lowest errors in each category are highlighted.

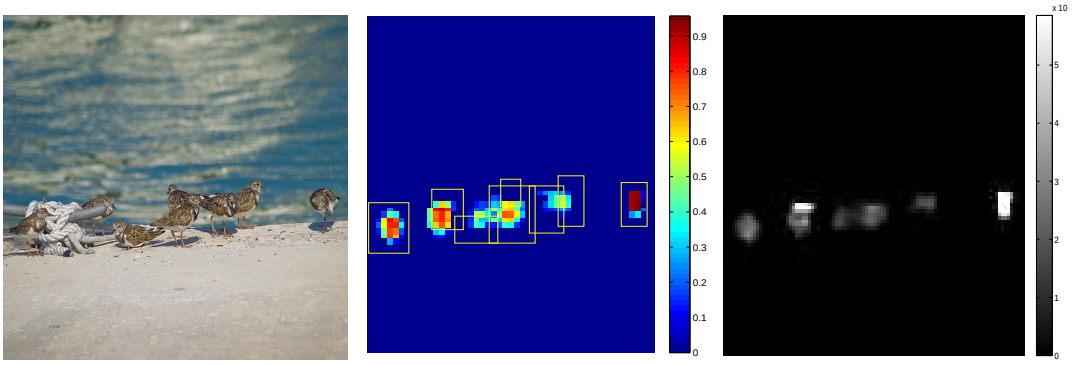


Figure 6.2: (left) a sample image from MS COCO "bird" category. (center) The classification scores after running a "bird" classifier on a 100×100 grid over the image. the ground truth bounding boxes in yellow. (right) Cox posterior mean intensity

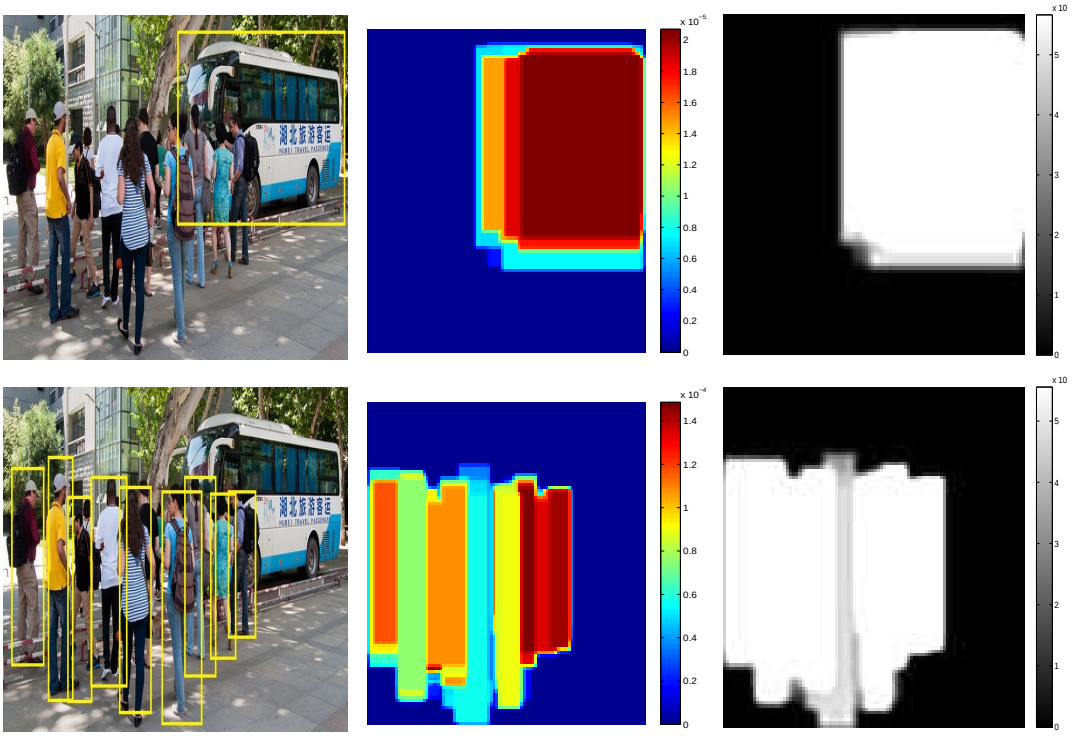


Figure 6.3: (left) A sample image from MS COCO containing 2 categories: "bus" and "person" as shown in yellow bounding boxes. (center) Input measurements for Cox computed using the region proposals from "Faster RCNN". (right) Cox posterior mean intensity for "bus"(top) and "person" (bottom)

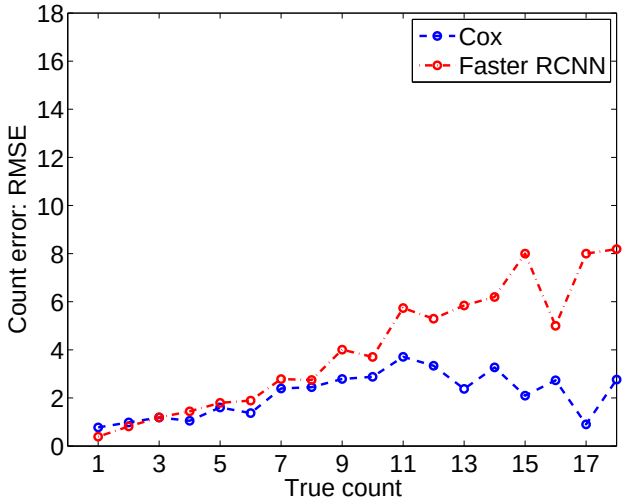


Figure 6.4: RMSE error plotted against object counts on the x-axis. The RMSE was computed for 3 categories: {*bird, sheep, motorbike*} and then averaged. The inputs y for Cox are the scores from a classifier run on a 100×100 grid over the image. Faster RCNN results shown in red for comparison on the same dataset. Cox has a lower error for all object counts, especially when the counts are larger.

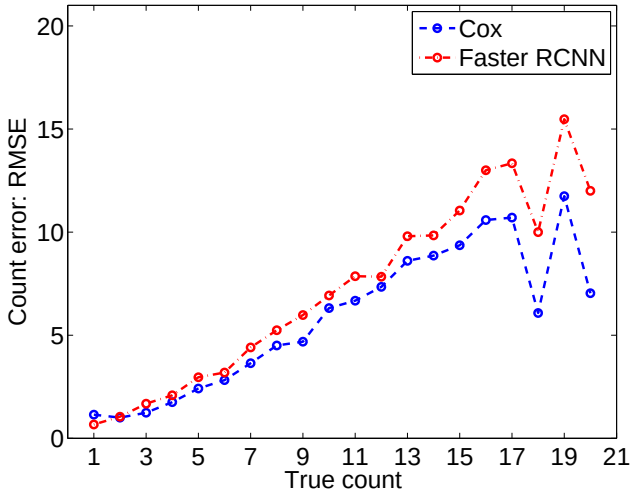


Figure 6.5: RMSE error plotted against object counts on the x-axis. The RMSE was computed for 20 object categories and then averaged. The inputs y for Cox are the scores from a FRCNN's region proposals. Faster RCNN results shown in red for comparison on the same dataset. Cox has a lower error for all object counts.

6.3.1 Hyper-parameter Estimation

In our experiments, we used the squared exponential kernel for the covariance matrix Σ. We notate the squared exponential kernel,

$$k_{SE}(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2l^2}\right) \tag{6.1}$$

We compute the parameters, (σ, l) of the kernel using the *method of moments* equations derived in chapter 4. They are reproduced below for reference.

$$\hat{\sigma}^2 = \frac{\bar{N}}{|\Omega|} \text{ and } \hat{l}^2 = \frac{|\Omega|\,(S^2 - \bar{N})}{2\pi\bar{N}^2} \tag{6.2}$$

where $|\Omega|$ is the size of the image; and $\bar{N}$ and S are the sample mean and sample standard deviation respectively. $\bar{N}$ and S were computed from the MS COCO [24] training dataset for each of the 20 classes mentioned in Table 6.1.

6.3.2 Evaluation

We use the *root-mean-square-error* (RMSE) as our evaluation metric for counting. For each category j, and given the prediction $\hat{c_{ij}}$ and ground truth c_{ij}, we compute RMSE as,

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(c_{ij} - \hat{c_{ij}})^2} \tag{6.3}$$

where N is the total number of images of category j in the dataset.

We report 2 flavors of RMSE in our experiments: (1) We compute the RMSE values on bootstrap samples for each object category and report the

mean and standard deviation over all bootstrap iterations. (2) Let k denote the number of object instances per image of category j. For each bootstrap sample we compute RMSE separately for each value of k, and then find the mean per sample and finally calculate the mean and variance over all bootstrap iterations. We refer to this as *RMSE-k*.

6.3.3 Results

We evaluated our approach on 20 classes of objects from MS COCO dataset. We follow two different ways of deriving the initial output for the Cox counter (see sections 6.2 and 6.2).

To get the first set of results, we use the region proposals from "Faster RCNN" as inputs to the Cox model; and then use the resulting intensity to estimate the counts. Fig. 6.3 illustrates this on an image containing two categories: "bus" and "person". The first column of this figure shows the original image with the ground truth bounding boxes in yellow. The input measurements for Cox computed from the region proposals are shown in the second column. The Cox posterior intensity is computed for each category separately. These results are shown in the last column of the same figure: top right image is the intensity for category "bus" and the bottom right one is the intensity for category "person".

Fig. 6.5 plots the RMSE values against object counts on the x-axis. For this plot, the RMSE values were computed for 20 object categories and then averaged. The inputs for Cox are the scores from a FRCNN's region proposals. "Faster RCNN" results shown in red for comparison on the same dataset. Cox

Method	bird	bike	sheep
FRCNN	2.00 (0.25)	1.37 (0.20)	2.06 (0.16)
Cox	**1.85** (0.07)	**1.13** (0.14)	**1.44** (0.07)

Table 6.3: RMSE bootstrap mean (std) for 3 object categories. The inputs y for Cox are the scores from a classifier run on a 100×100 grid over the image. The lowest errors in each category are highlighted.

has a lower error for all object counts.

Tables 6.1 and 6.2 show the RMSE and RMSE-k values for 20 categories in MS COCO validation dataset. These results use region proposals as Cox inputs. In all categories except "aeroplane", *Cox with region proposals* outperform "Faster RCNN" for counting, which demonstrates that our approach is promising.

The second set of experiments uses a subset of images from MS COCO validation set. These images belong to three categories: {bird, sheep, mbike}. The left-most image in Fig. 6.2 shows a sample image from MS COCO "bird" category. The classification scores after running a "bird" classifier on a 100×100 grid over the image are shown in the image in the center, along with the ground truth bounding boxes in yellow. The right-most image shows the corresponding Cox posterior mean intensity. For this set of experiments, we used the pre-trained bounding box classifiers from "Faster RCNN". Fig. 6.4 plots the RMSE values against object counts on the x-axis. Cox performs better than "Faster RCNN", especially when the object counts are larger. Tables 6.3 and 6.4 reports RMSE and RMSE-k values for this set of experiments.

Method	bird	bike	sheep
FRCNN	2.77 (0.29)	2.79 (0.42)	3.03(0.32)
Cox	**2.33** (0.12)	**1.86** (0.34)	**1.84** (0.12)

Table 6.4: RMSE-k bootstrap mean (std) for 3 object categories; computed separately for each value of object count k and then averaged. The inputs y for Cox are the scores from a classifier run on a 100×100 grid over the image. The lowest errors in each category are highlighted.

Fig. 6.6 plots the RMSE (root mean square) values against the object counts for three categories of objects: "bird", "sheep", "motorcycle". The number of objects per image in this dataset ranges from 1 to 18. The inputs for the Cox model were generated as described in section 3.3.2 of the main paper. The RMSE values for Cox counts are plotted in blue. "Faster RCNN's" results are shown in red for comparison on the same dataset. Cox has a lower error for all object counts in all categories.

Fig. 6.7 illustrates the details mapping the region proposals from "Faster RCNN" to a Cox intensity using an image containing two categories: "horse" and "person". The first column of this figure shows the original image with the ground truth bounding boxes in yellow. The input measurements for Cox computed from the region proposals are shown in the second column. The Cox posterior intensity is computed for each category separately. These results are shown in the last column of the same figure: top right image is the intensity for category "person" and the bottom right one is the intensity for category "horse". Note that the Cox intensity values and proposal scores are 0 when the image does not contain the category that we are looking for, as

shown in the third row of the same figure for the "motorcycle" category.

Finally, Fig. 6.8 shows the Cox posterior intensity and the final estimated count on two images in the "sheep" category. The second row has a fairly difficult image with both crowding and overlapping. The estimated count in this case is still very close to the true count.

These results show that combining region proposals from networks like "Faster RCNN" with Cox modeling significantly improves object count estimates. It also follows that this approach would potentially improve detection performance as well.

6.4 Cox Process for Localization

We show some preliminary results for object localization here. As described in Sect. 6.2, we replace the region proposal scores from "Faster RCNN" with Cox counts and continue "Faster RCNN" as-is for detection. Fig. 6.9 compares the detection results from Cox with "Faster RCNNN" on "bird" category. The image on the right(Cox) shows significant improvement in detection performance over the one on the left(Faster RCNN). This result demonstrates that using Cox modeling would offer improvements in localization as well as counting. Note that while we have used "Faster RCNN" as the baseline for counting and localization, this idea can be implemented on any network that uses a set of bounding boxes and scores in one of the intermediate steps. We intend to explore this in the next stages of the project.

6.5 Summary

In this chapter, we described a framework that improves the effectiveness of object counting and detection networks by converting the region proposals to Poisson intensity. This idea can be applied to any network that uses *proposal windows and object scores* in any of the intermediate layers. We demonstrated this approach on "Faster RCNN" and showed that there is a significant improvement in their counting performance as a result.

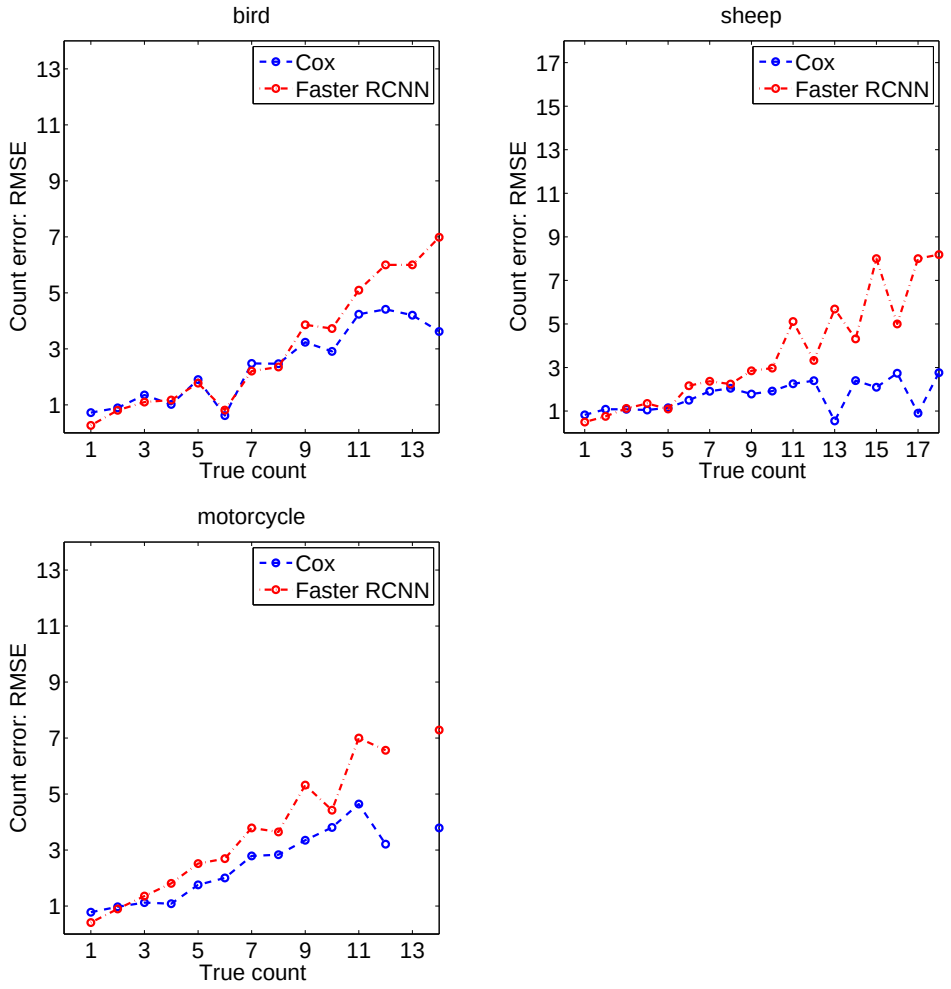


Figure 6.6: RMSE error(y-axis) plotted against object counts(x-axis) for three categories {"bird", "sheep", "motorcycle"}. The inputs for the Cox model are the scores from a bounding box classifier run on a 100×100 grid over the image. Faster RCNN results are shown in red for comparison on the same dataset. Cox has a lower error for all object counts in all three categories, especially when the counts are larger.

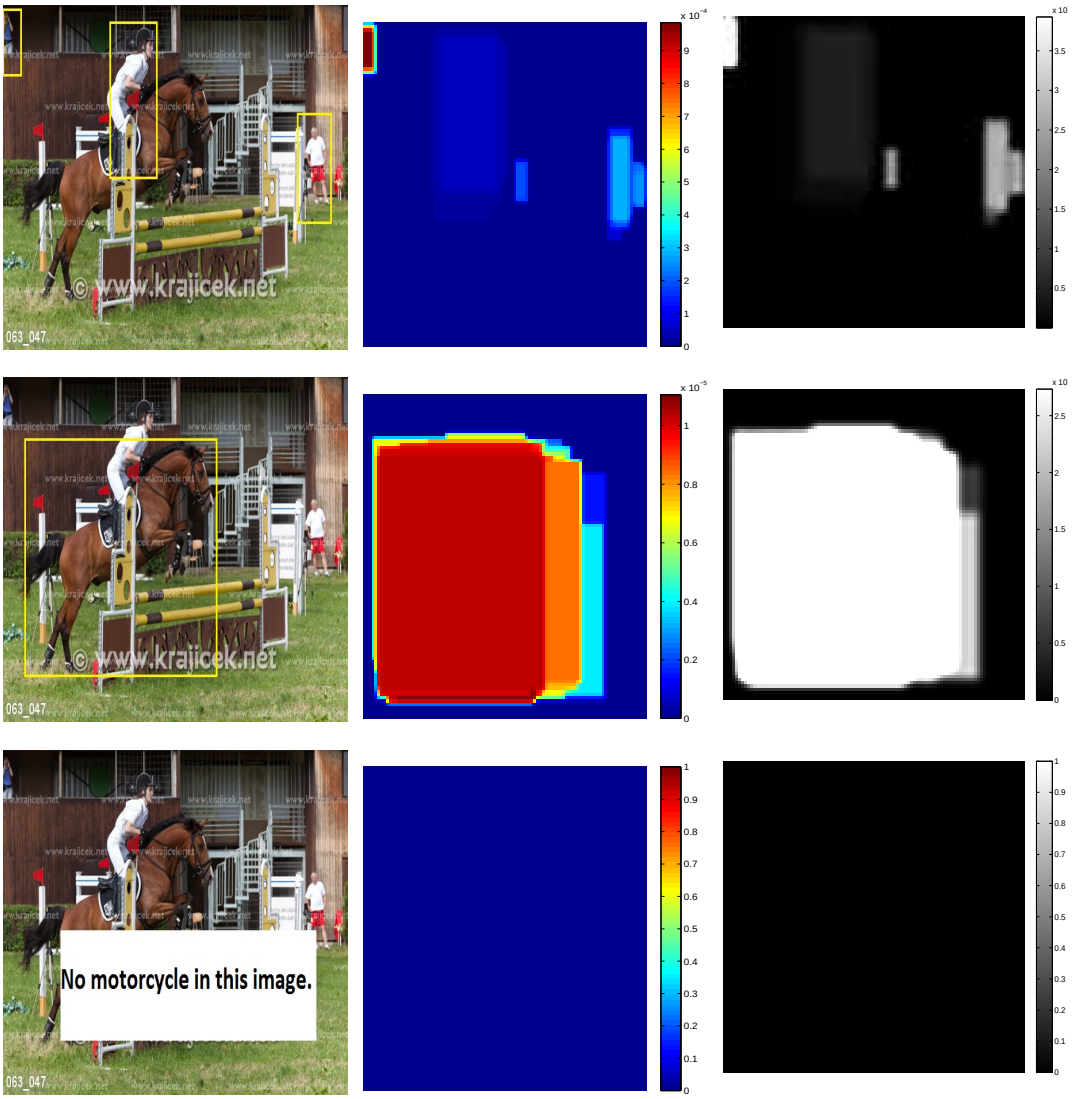


Figure 6.7: (left) A sample image from MS COCO containing two object categories: "horse" and "person". We run the RPN+Cox algorithm to count three types of objects: {"person", "horse", "motorcycle" } Note that this image does not contain a "motorcycle". **(center)** Input measurements for Cox computed using the region proposals from "Faster RCNN" for each of the three categories. **(right**) Cox posterior mean intensity for "person"(row1), "horse"(row2) and "motorcycle" (row3). Note that the Cox intensity values and proposal scores are 0 when the image does not contain the category that we are looking for (see row3).

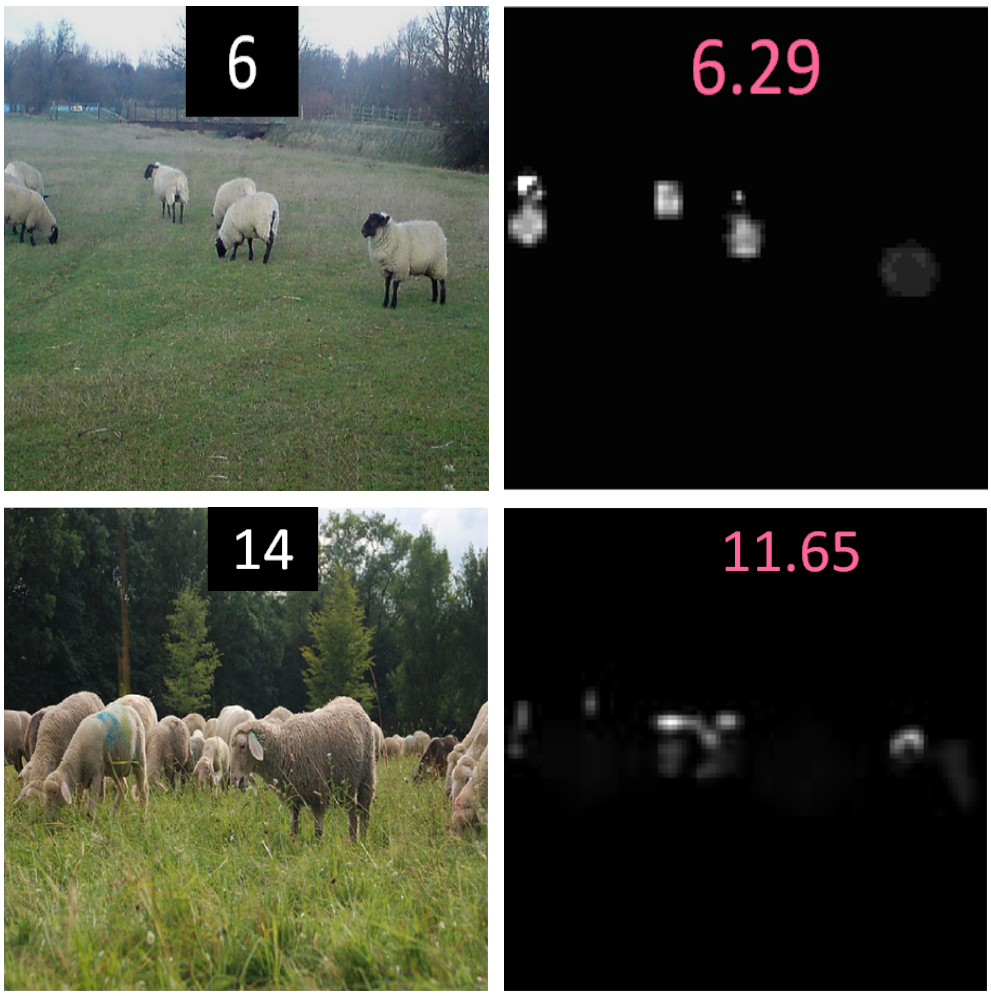


Figure 6.8: Counting sheep: (left) Original image with the true count overlaid. (right) Cox posterior intensity with the final estimated count shown in pink.

Figure 6.9: Preliminary localization results on an image from MS COCO "bird" category. **(left)** Detection result from "Faster RCNN" **(right)** Object detection after updating the "Faster RCNN's" region proposal scores using Cox. Here we replace the region proposal scores with Cox counts and continue "Faster RCNN" as-is for detection. Using the Cox model to re-map the regional proposal scores show an improvement in 'bird" localization in this example.

Part II

Object Localization

Chapter 7

Bayesian Multiple Object Localization

7.1 Introduction

In this part of the work, we consider the problem of localizing multiple instances of the same object using sub-region counts, exploring object localization from a Bayesian point of view. The search space is partitioned and on each sub-set, we ask questions of the form, “How many instances are there in this set?", while obtaining noisy answers. This setting is a generalization of the game of 20 questions to multiple targets. In general, we assume that the targets or instances are points on the real line, or in a two dimensional plane for the experiments, drawn independently from a known distribution.

We present the problem in one dimension for simplicity: Let $\Omega = \mathbb{R}$ be the real line and $\theta = (\theta_1, \ldots, \theta_k) \in \Omega^k$ be a vector containing the unknown locations of k objects, where $k \geq 1$ is known. One can sequentially choose subsets $A_1, A_2, \ldots$ of Ω, query the number of objects in each set, and obtain a series of noiseless answers $Z_1, Z_2, \ldots$. Our goal is to devise a method for

choosing the questions that allows us to find θ as accurately as possible, given a finite budget of questions. We work in a Bayesian setting, and use the entropy of the posterior distribution on θ to measure accuracy.

Generally, literature considering such problems falls into two categories: those that consider a single target ($k = 1$) and those that consider multiple targets ($k \geq 1$). For single-target localization, [69] considered a Bayesian setting and used the entropy of the posterior distribution to measure accuracy, as we do here. Within this context, a number of policies have been proposed, such as the dyadic policy and the greedy probabilistic bisection [70], which was further studied in [71, 72]. [73] more recently generalized this probabilistic bisection policy to multiple questioners as well. A discretized version of probabilistic bisection was studied by [74].

For multiple-target localization, three variations appear frequently: the Group Testing problem [75, 76, 77, 78, 79], the subset-guessing game associated with the Random Chemistry algorithm [80, 81] and the Guessing Secret game [82]. In each case, the goal is to query subsets, A, of the search space to determine an unknown set S. In the Group Testing problem, questions are of the form: "Is $A \cap S \neq \emptyset$?" In the subset-guessing game associated with the Random Chemistry algorithm, questions are of the form "Is $S \subset A$?" In the Guessing Secret game, when queried with a set A, the responder chooses an element from S according to any self-selected rule and specifies whether this chosen element is in A. The chosen element itself is not revealed and may change after each question. Thus, the answer is 1 when $S \subset A$, 0 when $A \cap S = \emptyset$, and can be 0 or 1 otherwise.

Four major practical considerations, however, severely limit the usability of existing theoretical results in real applications. First, when multiple targets need to be located, significant noise in the query answers are typically observed in real applications but ignored in most theoretical analyses. Second, existing theoretical analyses lack generality, considering very specific models for what is observed, rather than a general observational model that could be adapted to the application at hand. Third, many methods with theoretical guarantees on query complexity require a great deal of computation, and cannot be used in computation-constrained applications. Fourth, existing theoretical analysis often do not make clear the computational gain possible over repeatedly applying optimal strategies for single-target localization, making simpler strategies based on localizing single-targets more attractive.

This work addresses these concerns, by proposing and then analyzing the dyadic policy for simultaneous localization of multiple targets. This policy and our analysis uses an observational model that allows noise, and is general enough to subsume Group Testing, Random Chemistry, and a wide variety of other problems. We provide an explicit expression for the expected entropy of the posterior after N queries from this policy, and together with a simple information-theoretic lower bound on the expected entropy under the optimal policy, we show an approximation guarantee for the expected entropy reduction under the dyadic policy. Using this result, we can then demonstrate significant computation gains over repeated single-target optimal localization. The dyadic policy can be computed quickly and is non-adaptive, making it easy to parallelize, and far simpler to implement than dynamic strategies.

Moreover, it allows easy and exact computation of the expected number of targets at each location of our space.

7.2 Localization: Theoretical Formulation

Let $\theta = (\theta_1, \ldots, \theta_k)$ be a random vector taking values in $\mathbb{R}^k$. θ_i represents the location of the ith target of interest, $i = 1, \ldots, k$. We assume that $\theta_1, \ldots, \theta_k$ are i.i.d. with density f_0, and joint density $p_0(\theta) = \prod_{i=1}^{k} f_0(\theta_i)$. We assume f_0 is absolutely continuous with respect to the Lebesgue measure and has finite differential entropy, which is defined in (7.3). We refer to p_0 as the Bayesian prior probability distribution on θ. Note that even if the targets are indistinguishable, they are modeled as a vector, and not a set. This is a key requirement for simplifying the combinatorics of the probabilistic analysis. We will ask a series of $N > 0$ questions to locate $\theta_1, \ldots, \theta_k$, where each question takes the form of a subset of $\mathbb{R}$. The answer to this question is the number of targets in this subset. However, this answer is not available to the questioner. Instead, a noisy version of this answer is available. More precisely, for each $n \in \{1, 2, \ldots, N\}$, the n^{th} question is $A_n \subset \mathbb{R}$ and its noiseless answer is

$$Z_n = \mathbb{1}_{A_n}(\theta_1) + \cdots + \mathbb{1}_{A_n}(\theta_k), \tag{7.1}$$

where $\mathbb{1}_A$ is the indicator function of the set A. The noisy observable answer is a random function of Z_n, namely

$$X_n = h(Z_n, W_n) \tag{7.2}$$

where h is a known function and W_n is a collection of independent random variable, which are also independent of θ. Note that our choice of the set A_n may depend upon the answers to all previous questions. Thus, the set A_n is random.

We call a rule for choosing the questions A_n a *policy*, and indicate it with the notation π. The distribution of A_n thus implicitly depends on π. When we wish to highlight this dependence, we use the notation P^π and E^π to indicate probability and expectation respectively. However, when the policy being studied is clear, we simply use P and E . We let Π be the space of all policies.

Throughout the paper, we use the notation $X_{a:b}$ for any $a, b \in \mathbb{N}$ to indicate the sequence $(X_a, \ldots, X_b)$ if $a \leq b$, and the empty sequence if $a > b$. We define $\theta_{a:b}$ and $A_{a:b}$ similarly.

We refer to the posterior probability distribution on θ after n questions and answers as p_n, so p_n is the conditional distribution of θ given $X_{1:n}$ and $A_{1:n}$.

After we exhaust our budget of N questions, we measure the quality of what we have learned via the differential entropy $H(p_N)$ of the posterior distribution p_N on the targets at this final time,

$$\begin{aligned} H(p_N) &= -E[\log p_N] \\ &= -\int_{\mathbb{R}^k} p_N(u_{1:k}) \log(p_N(u_{1:k}))\, du_{1:k}. \end{aligned} \tag{7.3}$$

Throughout this paper, we use " log " to denote the logarithm to base 2. We let $H_0 = H(p_0)$, and we assume $-\infty < H(p_0) < +\infty$. The posterior distribution p_N, as well as its entropy $H(p_N)$, are random for $N > 0$, as they

depend on $X_{1:N}$. Thus, we measure the quality of a policy $\pi \in \Pi$ as

$$R(\pi, N) = E^{\pi}[H(p_N)]. \tag{7.4}$$

Our goal is to characterize the solution to the optimization problem

$$\inf_{\pi \in \Pi} R(\pi, N). \tag{7.5}$$

Any policy that attains this infimum is called *optimal*. According to this definition, an optimal policy may not exist.

Beyond theoretical interest, a policy for which $H(p_N)$ is small is of practical interest. It was shown in (Jedynak et al. 2012), section 4.3, that an optimal policy allows for localizing θ efficiently in the case of $k = 1$. We conjecture that the same occurs for arbitrary values of k. While (7.5) can be formulated as a partially observable Markov decision process (Frazier, 2010), and can be solved, in principle, via dynamic programming, the state space of this dynamic program (which is the space of posterior distributions over θ) is too large to allow solving it through brute-force computation. Thus, rather than attempting to compute the optimal policy, we provide an easily computed lower bound on (7.5), and then study a particular policy, called the dyadic policy and defined below, whose performance is close to this lower bound. The dyadic policy is non-adaptive, which means that the choice of the n^{th} query region is independent of the previous $(n-1)$ answers. We will see below that the dyadic policy attains the infimum in Eq. 7.5 for $\pi \in \Pi_N$, and thus is optimal among non-adaptive policies. We will also see that its performance comes within a factor of two of the infimum for $\pi \in \Pi$, showing

that it is a two-approximation among adaptive policies.

7.3 Lower Bound on the Expected Entropy

We first present an information-theoretic lower bound on the best expected entropy achievable, and a proof sketch.

Theorem 2.

$$H_0 - \log(k+1)N \leq H_0 - C_k N \leq \inf_{\pi \in \Pi} R(\pi, N) \tag{7.6}$$

The main arguments of the proof are as follows: First, at step n, the largest reduction in entropy that can be obtained in one question and on average occurs when the answer X_{n+1} and the targets θ have the largest mutual information given the history $I(\theta, X_{n+1}|X_{1:n})$, see Geman and Jedynak (1996). Second, since X_{n+1} depends on θ only through Z_{n+1} given $X_{1:n}$,

$$I(\theta, X_{n+1}|X_{1:n}) = I(Z_{n+1}, X_{n+1}|X_{1:n}). \tag{7.7}$$

Third, Eq. (7.7) is upper bounded by the *channel capacity*

$$C_k = \sup_q H\left(\sum_{z=0}^{k} q(z) f(.|z)\right) - \sum_{z=0}^{k} q(z) H\left(f(.|z)\right), \tag{7.8}$$

where q is a point mass function over the set {0,...,k} and $f(.|z)$ is the density, or point mass function of the noisy answer X_{n+1} given the noiseless answer Z_{n+1}. Since the noiseless answer Z_{n+1} is discrete and takes values in a set of size $k+1$, C_k is bounded above by $\log(k+1)$ providing the first inequality in (7.6).

In the noiseless case, both lower bounds in (7.6) are identical. Moreover,

they are not achievable. Indeed, at $N = 1$, the target locations are independent, and so the answer to the first question is Binomial, and must have an entropy no better than $H\left(\text{Bin}\left(k, \frac{1}{2}\right)\right) < \log(k+1)$. Moreover, as the expected entropy reduction is the sum of the expected entropy reduction at each question, the lower bound is not achievable for any N.

7.4 The Dyadic Policy

We now define an easy-to-compute policy, called the dyadic policy, and indicate it with the notation π_D. We first recall that the quantile function of θ_1 is

$$Q(p) = \inf\{u \in \mathbb{R} : p \leq F_0(u)\}, \tag{7.9}$$

where F_0 is the cumulative distribution function of θ_1, corresponding to its density f_0. Then, the *dyadic policy* entails choosing at step $n \geq 1$ the set,

$$A_n = \left(\bigcup_{j=1}^{2^{n-1}} \left(Q\left(\frac{2j-1}{2^n}\right), Q\left(\frac{2j}{2^n}\right)\right]\right) \bigcap \text{supp}(f_0), \tag{7.10}$$

where $\text{supp}(f_0)$ is the support of f_0, i.e., the set of values $u \in \mathbb{R}$ for which $f_0(u) > 0$. When f_0 is uniform over $(0, 1]$, the first question as per the dyadic policy is $A_1 = \left(\frac{1}{2}, 1\right]$, the second question is $A_2 = \left(\frac{1}{4}, \frac{1}{2}\right] \cup \left(\frac{3}{4}, 1\right]$, and each subsequent question is obtained by subdividing $(0, 1]$ into 2^n equally sized subsets, and including every second subset. A further illustration of the dyadic question sets A_n is provided in Figure 7.3. This definition of the dyadic policy generalizes a definition provided in Jedynak et al. (2012) for single targets.

The dyadic policy is easy to implement, and is non-adaptive, allowing its use in parallel computing environments. A detailed description of the *dyadic policy* is presented in the following sections. Section 7.4.1 explains the construction of the policy. In Section 7.4.2, we derive the rate of this policy and show that it is optimal among all non-adaptive policies.

7.4.1 Construction of the Dyadic Policy

The definition of the dyadic policy is given in Eq. (7.10). In this section, we provide an iterative construction of this policy, introducing notation which will be useful later on.

First, we partition the support of f_0 into two subsets, $A_{1,0}$ and $A_{1,1}$:

$$A_{1,0} = \left(Q\left(0\right), Q\left(\frac{1}{2}\right)\right] \cap \text{supp}(f_0), \tag{7.11}$$

$$A_{1,1} = \left(Q\left(\frac{1}{2}\right), Q\left(1\right)\right] \cap \text{supp}(f_0), \tag{7.12}$$

where Q, as defined in (7.9), denotes the quantile function. With this partition, the question asked at time 1 is:

$$A_1 = A_{1,1}. \tag{7.13}$$

Then we adopt a similar procedure recursively for each $n = 1, \ldots, N-1$ to partition $A_{n,j}$ into two subsets, $A_{n+1,2j}$ and $A_{n+1,2j+1}$ and then construct the question from these partitions. For $j = 0, \ldots, 2^n - 1$, define

$$A_{n+1,2j} = \left(Q\left(\frac{2j}{2^{n+1}}\right), Q\left(\frac{2j+1}{2^{n+1}}\right)\right] \cap \text{supp}(f_0), \tag{7.14}$$

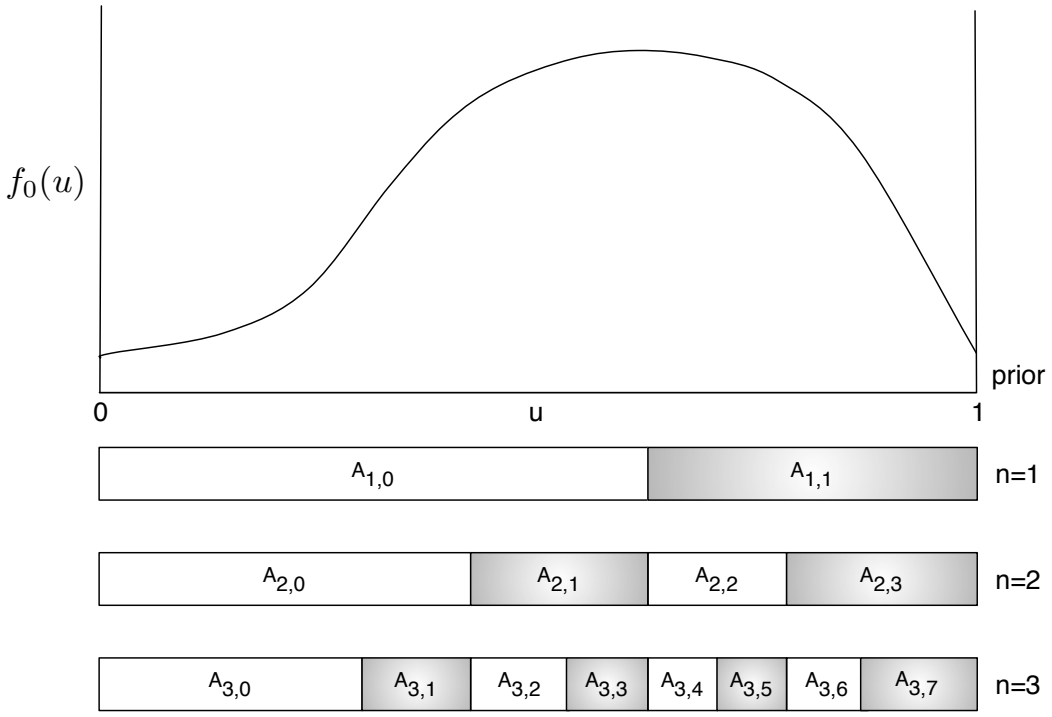


Figure 7.1: Illustration of the dyadic policy. The prior density with support $[0,1]$ is displayed above the illustrations of the sets $A_{n,k}$ for $n = 1,2,3$. The question set A_n is the union of the dark subsets $A_{n,k}$ for that value of n.

$$A_{n+1,2j+1} = \left(Q\left(\frac{2j+1}{2^{n+1}}\right), Q\left(\frac{2j+2}{2^{n+1}}\right)\right] \cap \text{supp}(f_0), \tag{7.15}$$

Then the question asked at time $n+1$ is

$$A_{n+1} = \bigcup_{j=0}^{2^n-1} A_{n+1,2j+1}. \tag{7.16}$$

An illustration of these sets A_n is provided below in Figure 7.1.

Note that the dyadic policy is non-adaptive, as only the prior distribution is used to construct the next set and not the answer to previous questions.

7.4.2 Expected Entropy under the Dyadic Policy

The following theorem provides an explicit expression for the expected entropy of the posterior distribution under the dyadic policy.

Theorem 3. *Under the dyadic policy π_D,*

$$R(\pi_D, N) = H_0 - D_k N, \tag{7.17}$$

where

$$D_k = H\left(\frac{1}{2^k}\sum_{z=0}^{k}\binom{k}{z} f(.|z)\right) - \frac{1}{2^k}\sum_{z=0}^{k}\binom{k}{z} H\left(f(.|z)\right). \tag{7.18}$$

In the noiseless case, this simplifies to

$$D_k = H\left(\text{Bin}\left(k, \frac{1}{2}\right)\right), \tag{7.19}$$

the entropy of a Binomial distribution $\text{Bin}(k, \frac{1}{2})$.

This result is easier to interpret in a discrete setting. Consider, as we will in Section 8, an image of $M \times M$ pixels containing k instances of an object, located at random, uniformly and independently. The instances are our targets. The starting entropy, neglecting the fact that several instances might occupy the same location, is

$$H_0 = k \log M^2. \tag{7.20}$$

According to (7.17), the expected number of questions N^* such that the k targets are located with certainty when using the dyadic policy, i.e, $R(\pi_D, N^*) = 0$, is such that

$$N^* = \frac{k}{D_k} \log M^2. \tag{7.21}$$

Consider the noiseless case for simplicity. Firstly, N^* is negligible compared to

the number of questions asked by a naive algorithm that queries the pixels in a fixed, predetermined order, for example line by line and column by column. This naive algorithm requires on average $\frac{M^2}{2}$ queries for a single target and more for more targets. Secondly, N^* is also better than querying optimally one target, which requires $\log M^2$ queries, and repeating this k times, for a total of $k \log M^2$ queries. Indeed,

$$N^* \sim \frac{2k}{\log \frac{\pi e k}{2}} \log M^2 \tag{7.22}$$

using the approximation of the Binomial distribution $B(k, \frac{1}{2})$ with the Normal distribution.

We can also compare the expression (7.17) for the expected entropy under the dyadic policy to the lower bound on the optimal expected entropy from Theorem 2. In both cases the expected entropy decreases linearly in the number of questions, with a reduction per question of D_k under the dyadic policy, and a reduction of C_k in the lower bound. This implies the following approximation guarantee for the entropy reduction under the dyadic policy, relative to optimal.

Corollary 1.

$$\frac{H_0 - R(\pi_D, N)}{H_0 - \inf_{\pi \in \Pi} R(\pi, N)} \geq \frac{D_k}{C_k}.$$

The approximation ratio D_k / C_k depends upon the noise model and the number of targets k.

In the noiseless case, this approximation ratio is $H(\mathrm{Bin}(k, \frac{1}{2})) / \log(k+1) \geq \frac{1}{2}$ (this inequality is derived in Appendix B.4), showing that the dyadic policy

is a 2-approximation in the noiseless case. Moreover, this approximation ratio approaches $1/2$ as $k \to \infty$, showing that the dyadic policy does not achieve the lower bound from Theorem 2 for large values of k. However, as previously noted, this lower bound is not achievable. The precise value of an optimal policy remains unknown.

In addition to the expected entropy $E^{\pi_D}[H(p_N)|p_0]$, we are also concerned with the actual entropy $H(p_N)$ that we obtain in a specific trial. It would be beneficial if the actual entropy did not deviate too much from its expected value. It turns out to be the case for the dyadic policy under the assumptions that the prior density f_0 is bounded from above. In fact, we can decompose the actual entropy $H(p_n)$ into a sum of two terms: the first term is a sum of i.i.d. random variables and the second term is a converging martingale. Utilizing this fact, we can derive Theorem 4, which provides almost sure convergence and asymptotic normality for $H(p_n)$.

Theorem 4. *Assume f_0 is bounded from above. Then under the dyadic policy,*

$$\lim_{N\to\infty} \frac{H(p_N)}{N} = -H\left(\text{Bin}\left(k, \frac{1}{2}\right)\right) \text{ almost surely,} \tag{7.23}$$

and

$$\lim_{N\to\infty} \frac{H(p_N) + NH\left(\text{Bin}\left(k, \frac{1}{2}\right)\right)}{\sqrt{N}} \overset{d}{=} N(0, \sigma^2), \tag{7.24}$$

where σ^2 is the variance of the random variable $\log\binom{k}{X}$ with $X \sim \text{Bin}\left(k, \frac{1}{2}\right)$.

Figure 7.2 below shows simulation results for localizing three instances under the dyadic policy. We assume the prior density f_0 is uniform over (0,1] and ask 100 questions. The first plot shows the entropy process $H(p_n)$. The

entropy reduction per question, which is visualized in the second plot, is asymptotically equal to $H\left(\text{Bin}\left(3, \frac{1}{2}\right)\right) = 1.8113$ according to the law of large numbers. The third plot illustrates the asymptotic normality of the entropy process for the dyadic policy.

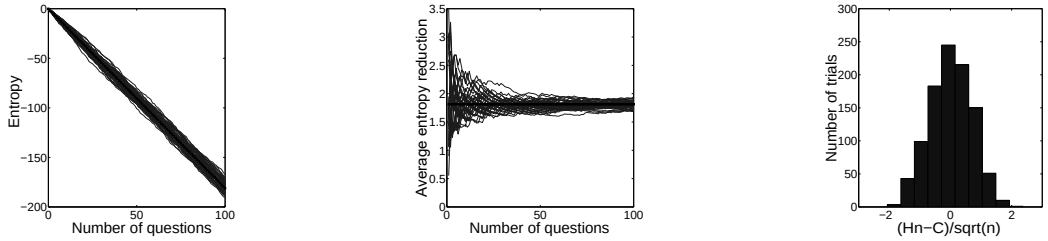


Figure 7.2: Simulation results for localizing three instances under the dyadic policy. $N = 100$ and f_0 is uniform over $(0, 1]$. The horizontal graphs above show the actual trajectories of entropy $H(p_n)$, the average entropy reduction per question $-\frac{H(p_n)}{n}$, and the asymptotic normality of $\frac{H(p_N)+NH\left(\text{Bin}\left(k,\frac{1}{2}\right)\right)}{\sqrt{N}}$, respectively.

7.5 Explicit Characterization of the Posterior Distribution

Our algorithms use the dyadic policy as a first phase procedure for deciding the order over pixels in which to call the oracle in computer vision applications. We now introduce some additional notation and derive an explicit formula for the posterior distribution over the targets θ given the history of the noiseless answers.

Consider a fixed n, where $1 \leq n \leq N$. For each binary sequence $s = \{s_1, \ldots, s_n\}$, define,

$$C_s = \left(\bigcap_{1 \leq j \leq n; s_j = 1} A_j\right) \bigcap \left(\bigcap_{1 \leq j \leq n; s_j = 0} A_j^c\right) \bigcap \text{supp}(f_0). \tag{7.25}$$

The collection $\mathcal{C} = \{C_s : C_s \neq \emptyset, s \in \{0,1\}^n\}$ provides a partition of the support of f_0. A history of n questions provides information on which sets C_s contain which targets among $\theta_{1:k}$. The regions A_n in Eq. 7.25 follow the dyadic policy as illustrated below in Fig. 7.3 where the prior distribution is uniform.

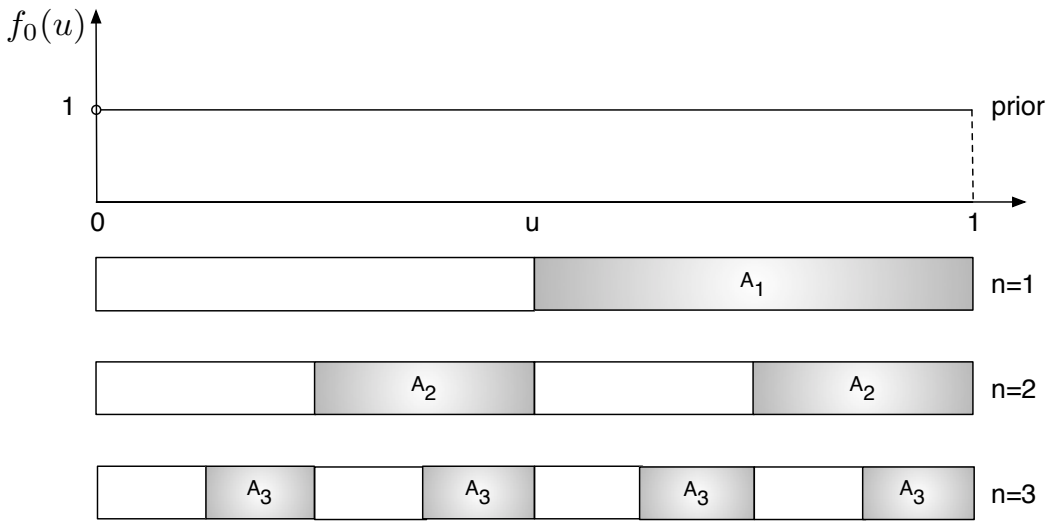


Figure 7.3: Illustration of the dyadic policy with uniform prior. The prior density f_0 displayed at top is uniform over (0,1]. The question set A_n is the union of the dark subsets for $n = 1, 2, 3$.

For each $C \in \mathcal{C}$, let $N(C) = \sum_{i=1}^{k} 1\{\theta_i \in C\}$ be the number of targets in C. We will think of a sequence of binary sequences $s^{(1)}, \ldots, s^{(k)}$ as a sequence of codewords indicating the sets in which each of the targets $\theta_{1:k}$ reside, i.e, indicating that θ_1 is in $C_{s^{(1)}}$, θ_2 is in $C_{s^{(2)}}$, etc. We may consider each binary sequence $s^{(1)}, \ldots, s^{(k)}$ to be a column vector, and place them into an $n \times k$ binary matrix, $\mathcal{S}$. This binary matrix then codes the location of all k targets, and is a codeword for their joint location.

Moreover, to characterize the location of the random vector $\theta = (\theta_{1:k})$ in

terms of its codeword $\mathcal{S}$, define $C_{\mathcal{S}} \subset \mathbb{R}^k$ to be the Cartesian product

$$C_{\mathcal{S}} = C_{s^{(1)}} \times \cdots \times C_{s^{(k)}}. \tag{7.26}$$

To be consistent with a noiseless answer Z_j, we must have exactly Z_j targets located in the question set A_j for each $1 \leq j \leq n$. This can be described in terms of a constraint on the matrix $\mathcal{S}$ as $s_j^{(1)} + \cdots + s_j^{(k)} = Z_j$, i.e., that the sum of the j^{th} row in the matrix $\mathcal{S}$ is X_j. Thus, given $\{Z_{1:n} = z_{1:n}\}$, the set of all possible joint codewords describing $\theta_{1:k}$ is

$$\begin{aligned} E_n = \{\mathcal{S} | s^{(1)}, \ldots, s^{(k)} \in \{0,1\}^n, C_s^{(1)}, \ldots, C_s^{(k)} \neq \emptyset, \\ s_j^{(1)} + \cdots + s_j^{(k)} = z_j, \text{for all } 1 \leq j \leq n\}. \end{aligned} \tag{7.27}$$

Now, we present the explicit characterization of the posterior distribution given $\{Z_1 \ldots Z_N\}$ in the following lemma, which will then be critical in letting us compute the posterior distribution after the dyadic questions, in Th 5.

Lemma 1.

$$p(u_{1:k}|Z_{1:N}) = \frac{p_0(u_{1:k})}{p_0\left(\bigcup\limits_{\mathcal{S} \in E_n} C_{\mathcal{S}}\right)}, \text{ for } u_{1:k} \in \bigcup_{\mathcal{S} \in E_n} C_{\mathcal{S}}, \tag{7.28}$$

where for a measurable set A, $p_0(A)$ denotes the integral $\int_A p_0(u_{1:k})\, du_{1:k}$. Moreover,

$$p_0\left(\bigcup_{\mathcal{S} \in E_n} C_{\mathcal{S}}\right) = \sum_{\mathcal{S} \in E_n} f_0(C_{s^{(1)}}) \ldots f_0(C_{s^{(k)}}), \tag{7.29}$$

where $f_0(C_{s^{(i)}})$ denotes the integral $\int_{C_{s^{(i)}}} f_0(u)\, du$.

We now provide a result that shows how to compute $E[N(C)|X_{1:N}]$, the expected number of targets within one of these sets C under the posterior

distribution. This result is important because this provides a means of ordering the image locations based on how likely they are to contain an object. This is used by the algorithms presented in chapter 8, at the start of a second phase in which an expensive noise-free oracle is called on some of the small sets C to establish definitively the number of targets in each. The algorithms use the value $E[N(C)|X_{1:N}]$ to determine the order in which to call the oracle.

Theorem 5. *For each instance θ_i and each $C \in \mathcal{C}$, the posterior likelihood, $P(\theta_i \in C|X_{1:N} = x_{1:N})$ satisfies*

$$P(\theta_i \in C|X_{1:N} = x_{1:N}) = \prod_{n=1}^{N} \left(\frac{e_n}{k}\right)^{s_n} \left(1 - \frac{e_n}{k}\right)^{1-s_n}. \tag{7.30}$$

where $e_n = E[Z_n|X_n = x_n]$ and $s_n = 1\{C \subseteq A_n\}$.

Moreover,

$$E[N(C)|X_{1:N} = x_{1:N}] = kP(\theta_i \in C|X_{1:N} = x_{1:N}).$$

The quantity e_n can be computed according to Bayes rule:

$$e_n = E[Z_n|X_n = x_n] = \sum_{j=0}^{k} jP(Z_n = j|X_n = x_n)$$

$$= \sum_{j=0}^{k} \frac{jP(X_n = x_n|Z_n = j)P(Z_n = j)}{P(X_n = x_n)},$$

where $Z_n \sim \text{Bin}\left(k, \frac{1}{2}\right)$, $P(X_n = x_n|Z_n = j)$ can be computed directly from the noise model (7.2), and $P(X_n = x_n) = \sum_{j=0}^{k} P(X_n = x_n|Z_n = j)P(Z_n = j)$.

7.5.1 Proof of Theorem 5: Posterior Ranking.

Proof. First, we prove the result for noiseless answers. Under the dyadic policy, we partition $(0, 1]$ into 2^N subintervals at time N. Now let us consider the event $\{\theta_i \in C | Z_{1:N} = z_{1:N}\}$, where C is one of such subintervals. Let us denote the support of the posterior distribution $p_N(u_{1:k})$ by $D = \bigcup_{\mathcal{S} \in E_N} C_{\mathcal{S}}$. Moreover, denote the collection of matrices $\mathcal{S} \in E_N$ that are consistent with the event $\{\theta_i \in C | Z_{1:N} = z_{1:N}\}$ by $E_N(C)$. Note that $p_0(C_{\mathcal{S}}) = f_0(C_{s^{(1)}}) f_0(C_{s^{(2)}}) \dots f_0(C_{s^{(k)}}) = 2^{-Nk}$ under the dyadic policy. Using Lemma 1, we can compute the probability of $P(\theta_1 \in C | Z_{1:N} = z_{1:N})$ as,

$$
\begin{aligned}
& P(\theta_1 \in C | Z_{1:N} = z_{1:N}) \\
&= \int_{u_{1:k} \in D_C} p_N(u_{1:k})\, du_{1:k} \\
&= \int_{u_{1:k} \in D_C} \frac{p_0(u_{1:k})}{\sum_{\mathcal{S} \in E_N} f_0(C_{s^{(1)}}) f_0(C_{s^{(2)}}) \dots f_0(C_{s^{(k)}})}\, du_{1:k} \\
&= \sum_{\mathcal{S} \in E_N(C)} \frac{1}{2^{Nk} |E_N|} \int_{u_{1:k} \in C_{\mathcal{S}}} p_0(u_{1:k})\, du_{1:k} \\
&= \frac{|E_N(C)|}{|E_N|},
\end{aligned}
\tag{7.31}
$$

where $|E_N(C)|, |E_N|$ denote the cardinalities of $E_N(C), E_N$, respectively. Consider the construction of the $N \times k$ binary matrix $\mathcal{S} \in E_N$. The only requirement it needs to satisfy is that the sum of n^{th} row is equal to z_n. Hence, we can construct the matrix row by row. Note that in step n, there are $\binom{k}{z_n}$ ways to specify the nonzero entries in the n^{th} row, for $n = 1, 2, \dots, N$. Thus, by the

product rule,

$$|E_N| = \prod_{n=1}^{N} \binom{k}{z_n}. \tag{7.32}$$

Using combinatorial techniques, we have

$$|E_N(C)| = \prod_{n=1}^{N} \binom{k-1}{z_n - s_n} 1_{\{0 \le z_n - s_n \le k-1\}}. \tag{7.33}$$

Combining (7.31), (7.32) and (7.33) together and using the fact that $\theta_1, \theta_2, \ldots, \theta_k$ are exchangeable,

$$\begin{aligned} P(\theta_i \in C | Z_{1:N} = z_{1:N}) &= \frac{\prod_{n=1}^{N} \binom{k-1}{z_n - s_n} 1_{\{0 \le z_n - s_n \le k-1\}}}{\prod_{n=1}^{N} \binom{k}{z_n}} \\ &= \prod_{n=1}^{N} \begin{cases} \frac{z_n}{k}, & \text{if } s_n = 1 \\ 1 - \frac{z_n}{k}, & \text{if } s_n = 0 \end{cases}, \end{aligned} \tag{7.34}$$

for $i = 1, 2, \ldots, k$.

Equivalently,

$$P\left(\theta_i \in C | Z_{1:N} = z_{1:N}\right) = \prod_{n=1}^{N} \left(\frac{z_n}{k}\right)^{s_n} \left(1 - \frac{z_n}{k}\right)^{1-s_n}. \tag{7.35}$$

Now we extend this result to the case with noisy answers. Firstly, we have

$$\begin{aligned} P(\theta_i \in C | x_{1:N}) &= \sum_{z_{1:N}} P(\theta_i \in C, z_{1:N} | x_{1:N}) \\ &= \sum_{z_{1:N}} P(\theta_i \in C | z_{1:N}, x_{1:N}) P(z_{1:N} | x_{1:N}) \\ &= \sum_{z_{1:N}} P(\theta_i \in C | z_{1:N}) P(z_{1:N} | x_{1:N}). \end{aligned} \tag{7.36}$$

Under the dyadic policy, $z_1, \ldots, z_N$ are conditionally independent given the noisy observations $x_1, \ldots, x_N$. Thus, $P(z_{1:N} | x_{1:N}) = \prod_{n=1}^{N} P(z_n | x_{1:N})$.

Moreover, due to the special structure of the dyadic policy, Z_n is independent of Z_j for all $j \neq n, j = 1, \ldots, N$; thus implying Z_n is independent of X_j for all $j \neq n, j = 1, \ldots, N$. Hence, $P(z_n|x_{1:N}) = P(z_n|x_n)$. Therefore, $P(z_{1:N}|x_{1:N}) = \prod_{n=1}^{N} P(z_n|x_n)$. According to (7.35), we have

$$\begin{aligned}
&P(\theta_i \in C|x_{1:N}) \\
&= \sum_{z_{1:N}} \left(\prod_{n=1}^{N} (\frac{z_n}{k})^{s_n} (1 - \frac{z_n}{k})^{1-s_n} \right) P(z_{1:N}|x_{1:N}) \\
&= \sum_{z_{1:N}} \prod_{n=1}^{N} (\frac{z_n}{k})^{s_n} (1 - \frac{z_n}{k})^{1-s_n} \prod_{n=1}^{N} P(z_n|x_n) \\
&= \sum_{z_{1:N}} \left(\prod_{n=1}^{N} (\frac{z_n}{k})^{s_n} (1 - \frac{z_n}{k})^{1-s_n} P(z_n|x_n) \right) \\
&= \prod_{n=1}^{N} \left(\sum_{z_n=0}^{K} (\frac{z_n}{k})^{s_n} (1 - \frac{z_n}{k})^{1-s_n} P(z_n|x_n) \right).
\end{aligned} \tag{7.37}$$

Furthermore, according to the definition of e_n, we have

$$\begin{aligned}
&\sum_{z_n=0}^{K} (\frac{z_n}{k})^{s_n} (1 - \frac{z_n}{k})^{1-s_n} P(z_j|x_n) \\
&= \begin{cases} \sum_{z_n=0}^{K} \frac{z_n}{k} P(z_n|x_n) = \frac{e_n}{k}, & \text{if } s_n = 1, \\ \sum_{z_n=0}^{K} (1 - \frac{z_n}{k}) P(z_n|x_n) = 1 - \frac{e_n}{k}, & \text{if } s_n = 0. \end{cases} \\
&= \left(\frac{e_n}{k} \right)^{s_n} \left(1 - \frac{e_n}{k} \right)^{1-s_n}
\end{aligned} \tag{7.38}$$

Substituting (7.38) into (7.37) proves the first claim in Theorem 5.

Finally,

$$E[N(C)|x_{1:N}] = \sum_{i=1}^{k} P(\theta_i \in C|x_{1:N}) = kP(\theta_i \in C|x_{1:N}), \tag{7.39}$$

and we complete the proof. □

7.5.2 Additive Gaussian Noise in the Screening Answers

In this section, we assume the noise model is given by $X_n|Z_n = Z_n + \epsilon_n$, where $\epsilon_1, \ldots, \epsilon_N$ are i.i.d. $\mathcal{N}(0, \sigma^2)$, where σ^2 is known. Then, according to the Bayesian formula, the expected answers $e_{1:N}$ conditional on the observations $x_{1:N}$ con be computed as,

$$\begin{aligned} e_n &= E[Z_n|X_n = x_n] \\ &= \sum_{j=0}^{K} jP(Z_n = j|X_n = x_n) \\ &= \sum_{j=1}^{K} j\frac{f_{\epsilon_n}(x_n - j)P(Z_n = j)}{\sum_{i=0}^{K} f_{\epsilon_n}(x_n - i)P(Z_n = i)} \\ &= \frac{\sum_{j=1}^{K} j \exp\left(-\frac{(x_n-j)^2}{2\sigma^2}\right)\binom{K}{j}}{\sum_{i=0}^{K} \exp\left(-\frac{(x_n-i)^2}{2\sigma^2}\right)\binom{K}{i}}, \end{aligned} \tag{7.40}$$

for each $n = 1, \ldots, N$. For example, when $\sigma = 1, K = 5, x_1 = 1$, we have $e_1 = E[Z_1|X_1 = 1] = 1.6317$. In general, $e_n = E[Z_n|X_n = x_n] \neq x_n$. A special case can be found when $x_n = \frac{K}{2}$. For example, when $\sigma = 1, K = 5, x_1 = 2.5$, then $e_1 = E[Z_1|X_1 = 2.5] = 2.5$.

7.5.3 Non-additive Noise in the Screening Answers

For the analysis in this section, we introduce some new notations. Define a binary random variable $Y_{i,n} = \mathbb{1}_{\theta_i \in A_n}$ to be the indicator of whether the i^{th} target is contained in the question set A_n, for all $i = 1, \ldots, K; n = 1, \ldots, N$. Then

$(Y_{i,n})_{n=1}^N$ encodes which set C, the target θ_i resides in. For each $i = 1, \ldots, K$, $Y_{i,1}, \ldots, Y_{i,N}$ follow i.i.d. Bernoulli($\frac{1}{2}$) and are conditionally independent given X_N. Thus, the posterior likelihood can be rewritten as,

$$\begin{aligned} P(\theta_1 \in C|X_N) &= P(Y_{1,1} = \mathbb{1}_{C\in A_1}, \ldots, Y_{1,N} = \mathbb{1}_{C\in A_N}|X_N) \\ &= \prod_{n=1}^{N} P(Y_{1,n} = \mathbb{1}_{C\in A_n}|X_N) \end{aligned} \tag{7.41}$$

Furthermore, since $Y_{i,n}$ is independent of each random observation X_m with $m \neq n$, the equation above reduces to

$$P(\theta_1 \in C|X_N) = \prod_{n=1}^{N} P(Y_{1,n} = \mathbb{1}_{C\in A_n}|X_n = x_n). \tag{7.42}$$

Now, consider a general non-additive noise model $X_n = f(Z_n, W_n)$ for $n = 1, \ldots, N$, where $W_1, \ldots, W_N$ are independent noise values. Let us focus on the iterated posterior ranking method. Suppose we have detected l instances (without loss of generality, assume these instances are $\theta_{K-l+1}, \ldots, \theta_K$) in the set C^* and denote this event by D. Using similar arguments, we have,

$$P(\theta_1 \in C|X_N, D) = \prod_{n=1}^{N} P(Y_{1,n} = \mathbb{1}_{C\in A_n}|X_n = x_n, D)$$

Hence, it suffices to compute $P(Y_{1,n} = 1|X_n = x_n, D)$.

Note that $P(Y_{1,n} = 1|X_n = x_n, D) = E[P(Y_{1,n} = 1|Z_n, X_n = x_n, D)|X_n = x_n, D]$. The inner conditional probability is either $(Z_n - l)/(k - l)$ if $\mathbb{1}_{C^*\in A_n} = 1$, i.e., if the known instances are in the set A_n, or $Z_n/(k - l)$ if $\mathbb{1}_{C^*\in A_n} = 0$, i.e., if the known instances are not in the set A_n. This can be written in both cases as $(Z_n - \mathbb{1}_{C^*\in A_n})/(k - l)$. Note that the information about $1_{C^*\in A_n}$ is contained in the event D.

Thus we have,

$$P(Y_{i,n} = 1|X_n = x_n, D) = (E[Z_n|X_n = x_n, D)] - \mathbb{1}_{C^* \in A_n})/(k - l)$$

It should be possible to compute this inner expectation tractably using Bayes rule. If $\mathbb{1}_{C^* \in A_n} = 0$, then it should be the same as $E[Z_n|X_n = x_n]$ in a case with $k - l$ instances. If $\mathbb{1}_{C^* \in A_n} = 1$, it is essentially the conditional expectation of the observation, given that there are at least l instances in the queried set.

7.5.4 Fast Implementation of the Posterior using Basis Images

Let x denote the input image that contains K objects and let $I = \{I_1, ..., I_N\}$ denote the basis images that correspond to the N questions. The basis images for a 16×16 image are illustrated in Figure 8.2. In the noiseless setting, we have from Eq. 7.35 that,

$$P\left(\theta_i \in C|Z_{1:N} = z_{1:N}\right) = \prod_{n=1}^{N} \left(\frac{z_n}{k}\right)^{s_n} \left(1 - \frac{z_n}{k}\right)^{1-s_n}.$$

Taking the logarithm on both sides we get,

$$\begin{aligned} \ln P(\theta_i \in C|z_{1:N}) &= \sum_{j=1}^{N} \ln \left((\frac{z_j}{K})^{s_j} (1 - \frac{z_j}{K})^{1-s_j} \right) \\ &= \sum_{j=1}^{N} s_j \ln(\frac{z_j}{K}) + (1 - s_j) \ln(1 - \frac{z_j}{K}) \end{aligned} \tag{7.43}$$

Setting $\alpha_j = \ln(\frac{z_j}{K})$ and $\beta_j = \ln(1 - \frac{z_j}{K})$, we have,

$$\ln P(\theta_i \in C | z_{1:N}) = \sum_{j=1}^{N} s_j \alpha_j + (1 - s_j)\beta_j \tag{7.44}$$

Note that,

$$s_j =< x_C, I_j > \text{ where } x_C \text{ denotes the location } C \text{ in the image } x \tag{7.45}$$

Substituting for s_j,

$$\begin{aligned}
\ln P(\theta_i \in C | z_{1:N}) &= \sum_{j=1}^{N} \big(< x_C, I_j > \alpha_j + (1 - < x_C, I_j >)\beta_j\big) \\
&= \sum_{j=1}^{N} \big(< x_C, I_j > (\alpha_j - \beta_j) + \beta_j\big) \\
&= \sum_{j=1}^{N} \big(< x_C, I_j > (\alpha_j - \beta_j)\big) + \sum_{j=1}^{N} \beta_j \\
&=< x_C, \sum_{j=1}^{N} I_j(\alpha_j - \beta_j) > + \sum_{j=1}^{N} \beta_j
\end{aligned} \tag{7.46}$$

This shows that the dyadic posterior can be computed using a series of element-wise multiplications between the *input image* and *basis images*, plus additions, and thus very efficient. The computational complexity is $O(n \log_2 n)$, where n is the number of pixels.

Chapter 8

Algorithms for Localization

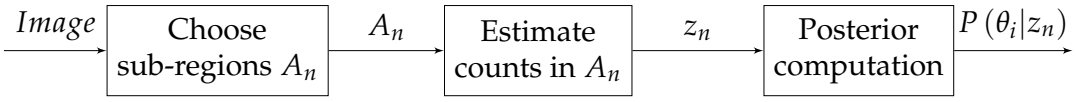


Figure 8.1: Block diagram for the localization work-flow.

We now show how the dyadic policy, analyzed above in the continuous setting using the entropy, can be used in an idealized computer vision setting to reduce the number of oracle calls required to locate k instances of a given object within a $M \times M$ digital image.

The dyadic policy is unique in the fact that it has a simple closed form expression for the posterior probability that the object instance is located at a pixel location C (see Theorem 5). We use the term "screening questions" to denote the instance count queries on the subset A. The "oracle" refers to the expensive but highly accurate classifier that will be run on a selected subset of the pixel locations. Note that instead of computing the entropy of the posterior distribution, we use the number of calls to the oracle as the measure

of performance.

In this setting, we use a $M \times M$ image in which the objects to be located are of size one pixel and have the intensity value 1 while the rest of the pixels are of intensity 0, with $k = \{2, 3, 10\}$ and $M = \{8, \ldots, 1024\}$. This simulation setting is far from a realistic computer vision application as the number of instances in each dyadic query set A will not be readily available and we would need to train an appropriate machine learning classifier that computes this value, albeit with noise ([42], [15]). Moreover, in realistic applications, the number of objects k will not be known in advance and the answers to the screening questions could be corrupted with noise. Nevertheless, this simulation experiment provides useful analysis of the performance of the dyadic policy and the algorithms that locate object instances using the posterior distribution computed from the answers to the screening questions using Theorem 5.

We consider algorithms that proceed in two phases, eventually iterated. In the first phase, we query the dyadic sets to find the instance count in each set, and compute the posterior distribution conditional on the dyadic observations. In contrast to the continuous domain, there is here a limited supply of dyadic sets in a discrete setting. Choosing for M a power of 2, there are $\log M$ dyadic horizontal queries and $\log M$ dyadic vertical queries (Figure 8.2 presents the dyadic questions for $M = 16$). In the second phase, we order the pixels based on how likely it is to find an object in them. We then query the oracle according to this ordering so as to find definite answers. We compare three algorithms: Posterior Rank (PR) Algorithm 4, Iterated Posterior

Rank (IPR) 5 and Entropy Pursuit (EP) 6. We will see that all these three algorithms significantly outperform the baseline algorithm – the Index Rank (IR) algorithm – in terms of the expected number calls to the oracle (see Figure 8.4). Note that IR scans the image left to right, top to bottom until it finds all instances.

The PR algorithm computes the expected number of instances $E[N(C)|X_{1:N}]$ in each pixel C using Theorem 5, orders the pixels in decreasing order of this quantity $E[N(C)|X_{1:N}]$, and runs oracle calls according to this order until all the instances are found. This algorithm is summarized below.

Algorithm 4 Posterior Rank (PR)

1: Compute the answers to the screening questions.
2: Compute the posterior rank r according to (7.30).
3: Run the oracle on the pixels according to r until all the instances are found.

The IPR Algorithm 5 is a variation of the PR Algorithm 4. As before, the pixels are searched in decreasing order of the expected number of instances. When the oracle locates an instance (or instances) at a pixel, the expected number of unlocalized instances at each pixel recomputed using Eq. (8.2) as described below, which provides an updated ranking for the remainder of the search.

Computing the expected number of unlocalized instances at a pixel, given the locations of $0 \leq i < k$ previously localized instances, is most straightforward in the case of additive noise, i.e.,

$$h(Z_n, W_n) = Z_n + W_n \tag{8.1}$$

In this case, this computation is accomplished by *masking* the instances already found, i.e., by subtracting those instances localized in A_n from X_n, subtracting the overall number of localized instances from k, and recomputing using (7.30). More generally, we re-use the expression (7.30), but replace the number of unlocalized instances k by $k' = k - i$, and alter e_n to account for those previously localized instances residing in the queried set A_n. Letting $N'(C) = N(C) - \sum_{j=1}^{i} 1\{\theta_j \in C\}$ be the number of unlocalized instances in C, we have

$$E[N'(C)|X_{1:N}, \theta_{1:i}] = k' \prod_{n=1}^{N} \left(\frac{e'_n}{k'}\right)^{s_n} \left(1 - \frac{e'_n}{k'}\right)^{1-s_n}. \tag{8.2}$$

Here, $e'_n = E[Z'_n|X_n = x_n, \theta_{1:i}]$, with $Z'_n = Z_n - \sum_{j=1}^{i} 1\{\theta_j \in C\}$ being the number of unlocalized instances in the queried set A_n. The quantity e'_n can be computed as,

$$e'_n = \sum_{j=0}^{k'} \frac{jP(X_n = x_n|Z'_n = j, \theta_{1:i})P(Z'_n = j)}{P(X_n = x_n|\theta_{1:i})},$$

where $Z'_n \sim \text{Bin}\left(k', \frac{1}{2}\right)$, $P(X_n = x_n|Z'_n = j, \theta_{1:i})$ can be computed from the noise model (7.2), and $P(X_n = x_n|\theta_{1:i}) = \sum_{j=0}^{k'} P(X_n = x_n|Z'_n = j, \theta_{1:i})P(Z'_n = j)$.

In the special case when no instances have been localized, so $i = 0$, (8.2) recovers (7.30).

We now summarize the IPR algorithm:

Algorithm 5 Iterated Posterior Rank (IPR)

1: Compute the answers to the screening questions.
2: **repeat**
3: Compute the posterior rank r according to (8.2).
4: Run the oracle on the pixels according to r until one or several instances are found at a pixel.
5: **until** all the instances are found.

IPR's Step 4 may request the oracle's feedback on a pixel that was already queried in a previous stage. This is because we condition on previously localized instances, but not on previous negative reports from the oracle that there were no instances at a particular pixel. When this occurs, we simply report the oracle's previous value, rather than re-running the oracle. We do not condition on all previous oracle results because this would make the computation of the posterior expected number of instances much more challenging.

Figure 8.2 and 8.3 illustrate the IPR algorithm for a 16×16 image with $k = 4$ instances. Figure 8.2 illustrates the screening questions under the dyadic policy, with light regions marking the questions sets. The first row of Figure 8.3 shows the true but unknown locations of the instances in each iteration of the IPR algorithm. The second row shows the expected number of instances within each pixel computed after screening questions in each iteration, respectively, with lighter regions having a higher expected number of instances.

Entropy Pursuit (EP) 6 is a greedy algorithm aimed at reducing the expected entropy on the joint location of the instances. It has been studied and used for locating and tracking instances in [83, 69, 84, 85, 86]. This algorithm

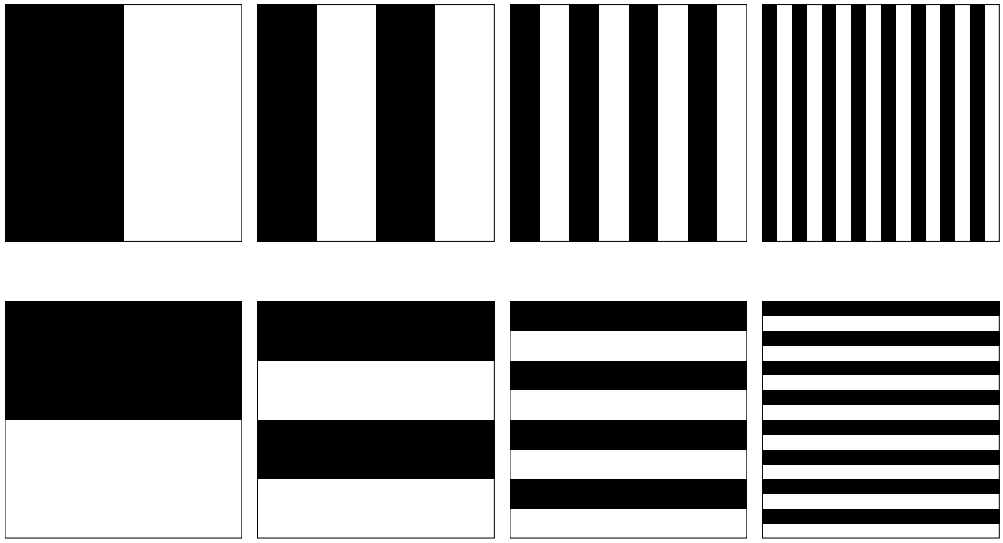


Figure 8.2: The queried regions under the dyadic policy for a 16×16 image shown in white.

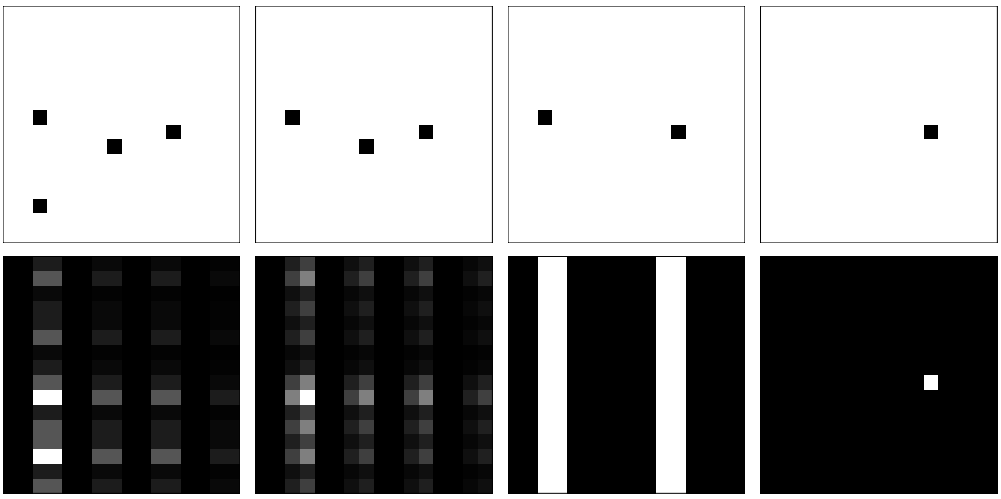


Figure 8.3: (row 1) Example image with 4 instances of the object initially, one instance is found after each iteration of the IPR algorithm. (row 2) The corresponding posterior distribution after each iteration. Light regions indicate pixels more likely to contain the object instance, while dark regions are less likely.

is closely related to the IPR algorithm. The differences between EP and IPR are: i) EP uses a different ordering criterion; ii) EP updates the ordering each time after running the oracle at a pixel instead of after an instance being found. Specifically, EP computes for each pixel, the expected entropy reduction – in the distribution of the location of the instances – that would potentially be achieved by running the oracle at this pixel. It then selects the pixel for which this quantity is maximal. A detailed implementation is given in the appendix (see B.6).

The EP algorithm is provided below.

Algorithm 6 Entropy Pursuit (EP)

1: Compute the answers to the screening questions.
2: Obtain E_N defined in (7.27), the collection of matrices characterizing possible joint instance locations.
3: **repeat**
4: Select the pixel for which the expected entropy reduction is maximum.
5: Run the oracle at this pixel.
6: Remove all the inconsistent matrices from the collection E_N.
7: **until** all the instances are found.

We use simulations to compare the performances of the three algorithms described above with a baseline algorithm, called Index Rank (IR). IR sweeps the image from left to right, top to bottom, until all the instances of the object are found. For the sake of simplicity, the object to be found in our simulation is a dot of size 1 pixel. In our simulation experiments, we use 100 random assignments for the locations of the object instances for each k and each image size, and measure the number of calls to the oracle required in each case.

8.1 Noiseless Answers to the Queries

We consider first the situation where the answers to the screening questions are noiseless, which is consistent with the theoretical analysis presented in the previous sections.

Figure 8.4, top row compares the algorithms for $k = 2$, $k = 3$ and $k = 10$ object instances for image sizes $\{8 \times 8, 16 \times 16, \ldots, 1024 \times 1024\}$. Algorithms PR 4, IPR 5 and EP 6 require a smaller average number of calls to the oracle compared to the baseline IR. An example will show how dramatic this is for large images: in the case of 1024×1024 pixel images and $k = 2$ instances, IR requires 2^{20} evaluations of the oracle while IPR requires less than 2^8 on average. As is evident from the plots, IPR is also much more efficient than PR. IPR and EP show similar performances, however, IPR is superior to EP in terms of the computational complexity. Due to the EP algorithm's large computational and memory requirements, we have only plotted EP for $k = 2$ and $k = 3$, and have only gone up to 512×512 image for $k = 3$.

8.2 Noisy Answers to the Queries

In an actual computer vision setting, screening questions would be answered by an image processing algorithm, trained using labeled data. These answers would then be noisy. We performed experiments to measure the effectiveness of the Posterior Rank (PR) and the Iterated Posterior Rank (IPR) algorithms in this case, while the oracle is still considered perfect. We use the additive model presented in (8.1) and we choose W_n to be independent, normally distributed

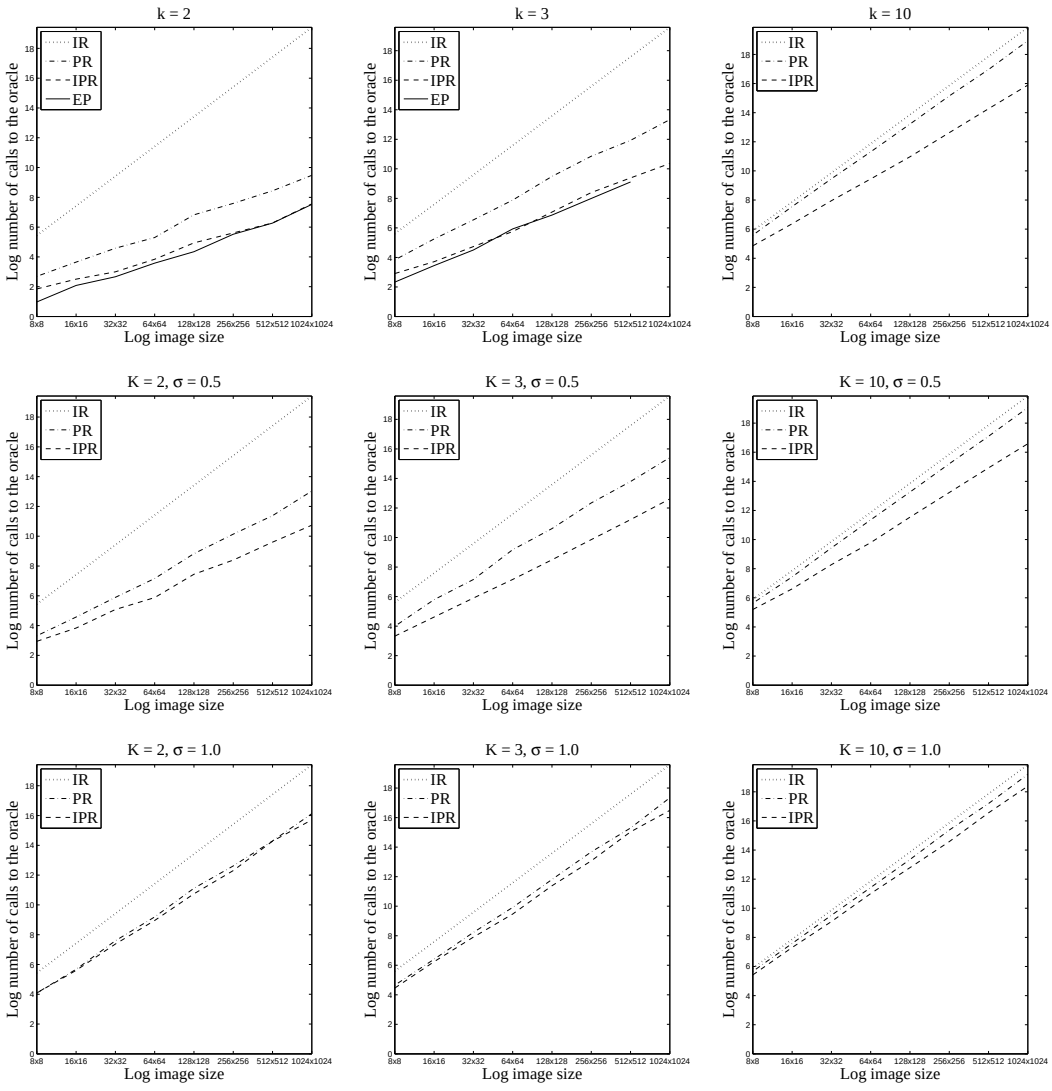


Figure 8.4: The mean number of calls to the oracle over 100 samples plotted against the image size for $k = 2$, $k = 3$ and $k = 10$ object instances using the algorithms 4, 5 and 6 described in chapter 8. (row 1): No noise in the screening answers. (row 2): Gaussian noise with $\sigma = 0.5$ in the screening answers. (row 3): Gaussian noise with $\sigma = 1.0$ in the screening answers.

random variables with standard deviation σ. Figure 8.4, second and third row compares the Posterior Rank (PR), Iterated Posterior Rank (IPR) and the Index Rank (IR) algorithms for $k = 2$, $k = 3$ and $k = 10$ object instances for two levels of noise ($\sigma = 0.5, 1$). We note that both algorithms outperform the default IR algorithm in all cases. The IPR algorithm is more robust to noise than the PR algorithm. As expected, the performances of both algorithms decrease as the amount of noise increases.

8.3 Object Detection

To illustrate the potential benefits of PR (Alg. 4) and IPR (Alg. 5) in a real world setting, we evaluate their performance in the context of finding faces in images. In particular, we consider how these strategies can be used as cost effective ways to determine regions where objects may potentially be located so that high-performing (and computationally expensive) classifiers may then be more effectively be used.

We begin by training an extremely efficient but poorly-performing face classifier, with the intention of evaluating it at each location of an image. To this end, we train a boosted classifier, as described in [28], but only with 50 stumps(*i.e.* 5% compared to state-of-the-art classifiers), using 4000 faces and 5 million background samples of size 30×30. Once trained, we evaluated our poor-classifier at multiple scales on 35 images from the MIT+CMU face dataset. For each image location, a pixel was scored as the sum of weighted stump outputs of our trained classifier (*e.g.* Fig. 8.5 (2nd row) illustrate these scores: higher values in white and lower values in black). From these response

image, we calculated the answers to the screening questions using an integral image representation followed by the PR algorithm (Alg. 4)

Fig. 8.5 depicts, for 2 images with 4 faces each, the corresponding response maps for the poor classifier and the posterior distribution after a new face has been located using the PR algorithm (*i.e.* higher values in white and lower values in black). Table 8.1 shows the average number of oracle calls required to find all the faces in the image.

The results for the IPR algorithm are shown in Fig. 8.6 and 8.7. Both figures show the original input image, the result from the poor-classifier in the first row. This is followed by the posterior image at each stage of IPR algorithm. Light regions indicate pixels more likely to contain the face, while dark regions are less likely. The location of the face detected at each stage is shown as a blue rectangle.

Table 8.1: Results for Posterior Rank algorithm on a dataset with 35 images containing 130 faces in total.

No. of Pixels	Oracle Calls	Oracle Calls per face
12713984	347238	2671

8.3.1 Experiments using a Binary Counter

For the dyadic localization to work well, we need fairly precise estimates of the number of instances in each dyadic region. However, it is challenging to build classifiers that calculate the exact count of object instances in all dyadic sub-regions of the image – this is especially true for the latter screening regions that are made up of thin strips that are only a few pixels wide. In this section,

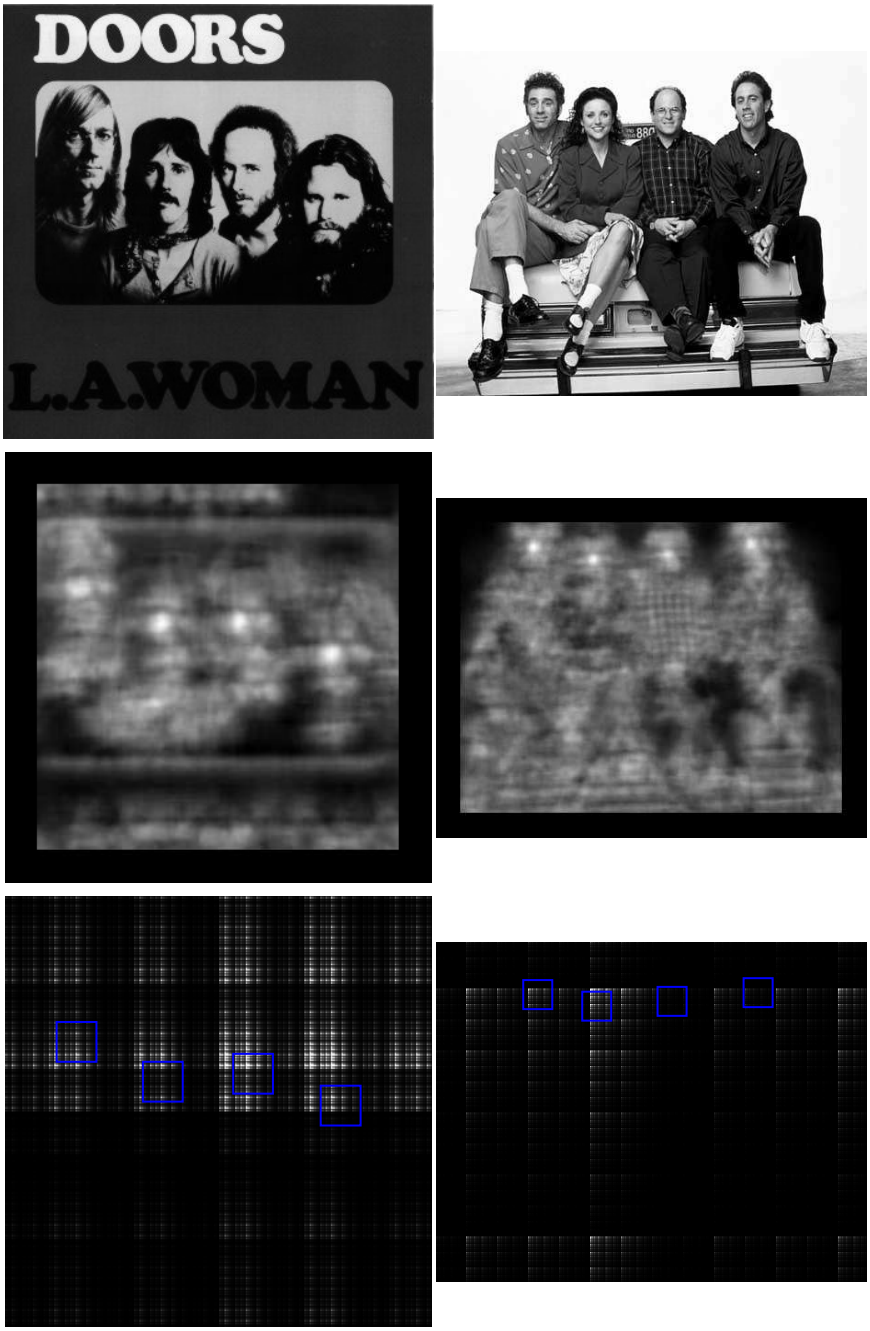


Figure 8.5: PR Face detection result. (row 1): 2 example images with 4 faces each. (row 2): Result from the poor-classifier. row(3): The posterior at found by the PR algorithm. Light regions indicate pixels more likely to contain the face, while dark regions are less likely. The locations of the faces detected are indicated as a blue rectangles.

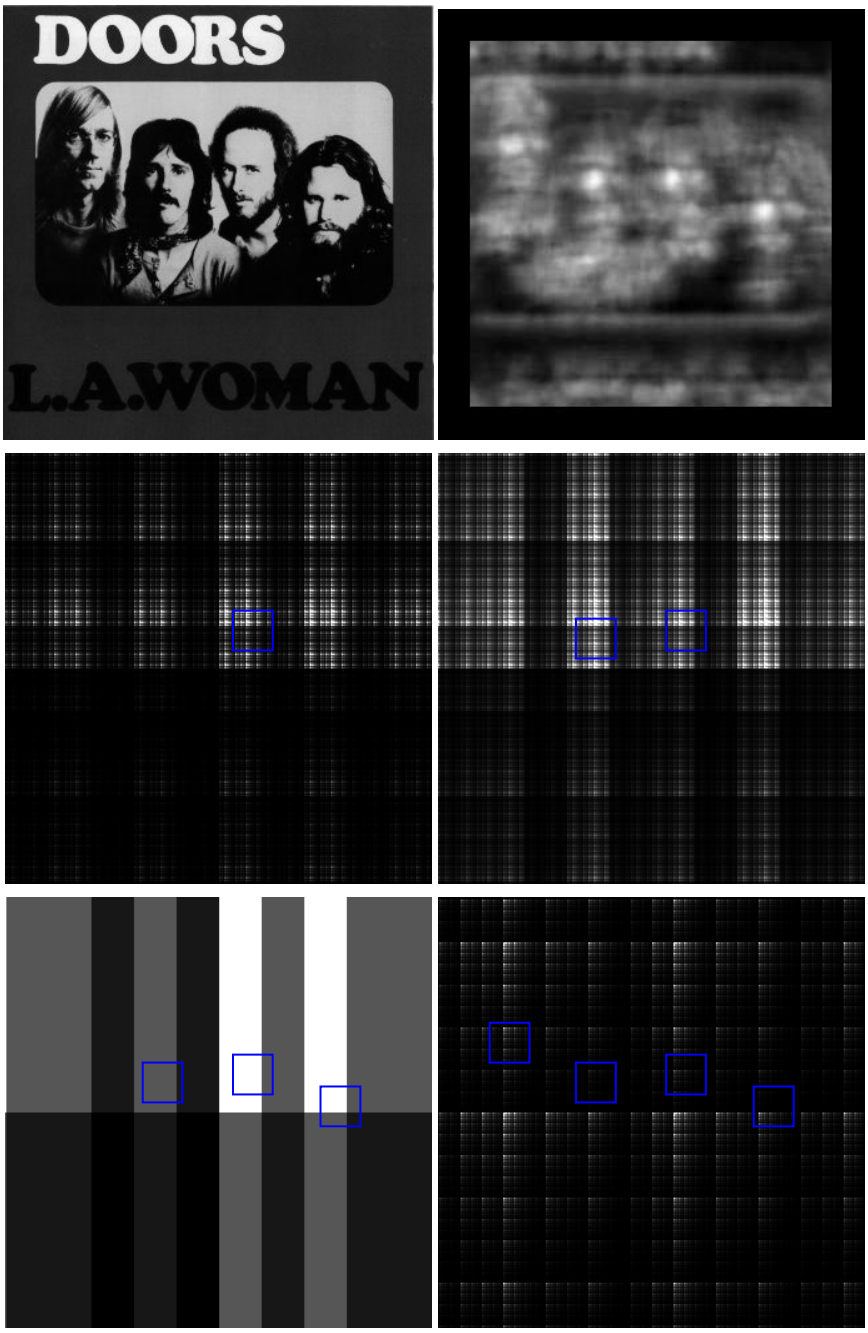


Figure 8.6: IPR Face detection result. (row 1): Left: An image with 4 faces. Right: Result from the poor-classifier. row(2,3): The posterior at each stage of IPR algorithm. Light regions indicate pixels more likely to contain the face, while dark regions are less likely. The location of the face detected at each stage is shown as a blue rectangle.

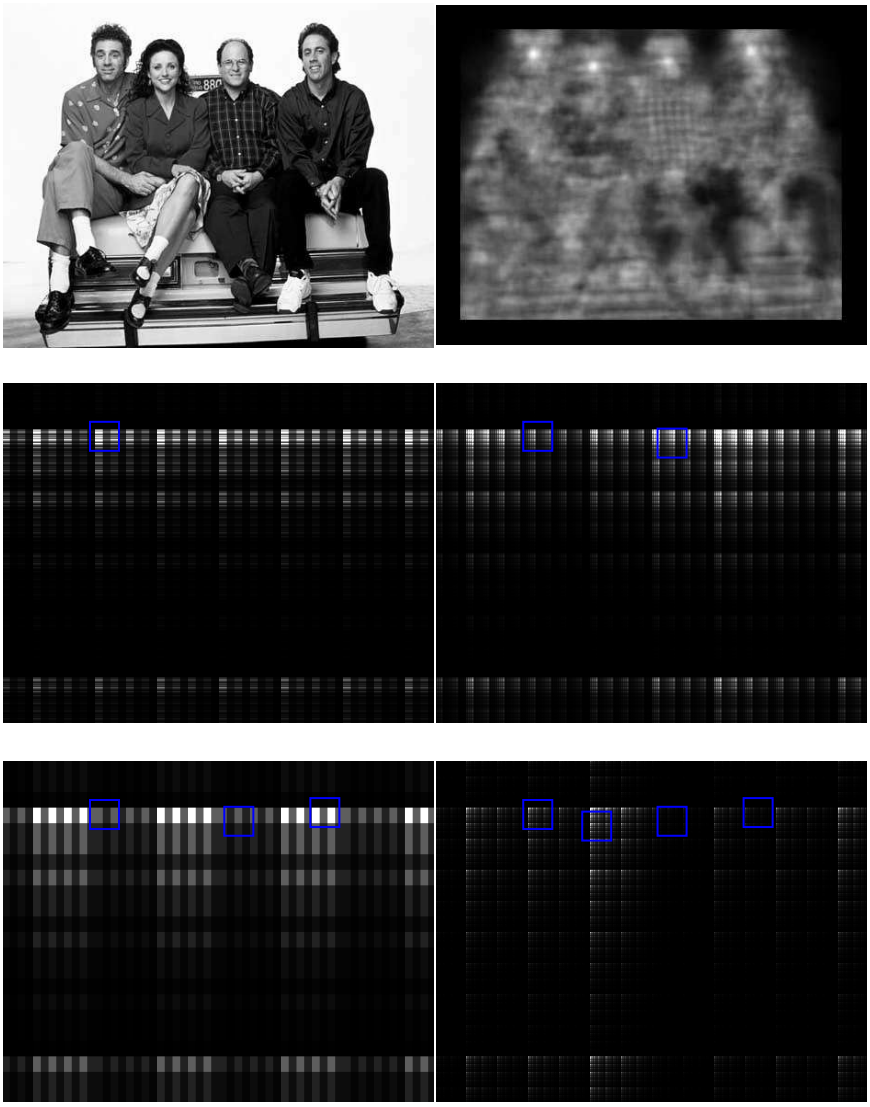


Figure 8.7: IPR Face detection result. (row 1): Left: An image with 4 faces. Right: Result from the poor-classifier. row(2,3): The posterior at each stage of IPR algorithm. Light regions indicate pixels more likely to contain the face, while dark regions are less likely. The location of the face detected at each stage is shown as a blue rectangle.

we explore the performance of the localization algorithms in the context of binary counters, *i.e.* counters that answer 1 if there is at least one object in the dyadic region and 0 otherwise. To this end, we train a Support Vector Machine (SVM) classifier using histogram intersection kernel. This classifier differentiates between regions having 0 faces and at least 1 face. The histogram on the poor-classifier response in each region A_n and its complement is used as the feature for classification. Fig. 8.8 provides an example of this. The second row of Fig. 8.8 shows the poor-classifier response for the first screening region and the corresponding normalized histogram. The results are shown in Table 8.2.

8.4 Augmenting the Dyadic Set

We noted in section 7.4.2 that the starting entropy H_0 for an image of size $M \times M$ containing k instances of an object, located at random, uniformly and independently is $k \log M^2$. The number of bits we can learn per question is limited by the channel capacity C_k. In order to fully locate all the instances of the object, we need $N = \frac{H_0}{C_k}$ questions. The channel capacity C_k is bounded from above by the capacity of a noiseless channel, *i.e.* $C_k \leq \log(k+1)$. This means that the number of questions required for any policy on average is, $N \geq \frac{k \log M^2}{\log(k+1)}$. The number of questions available for the dyadic policy is $\log M^2$. Since there is a clear need for more questions that cannot be satisfied by the dyadic policy, we explore ideas from *information theory and coding*, leading us to re-define the object localization problem as the problem of communication via a noisy channel.

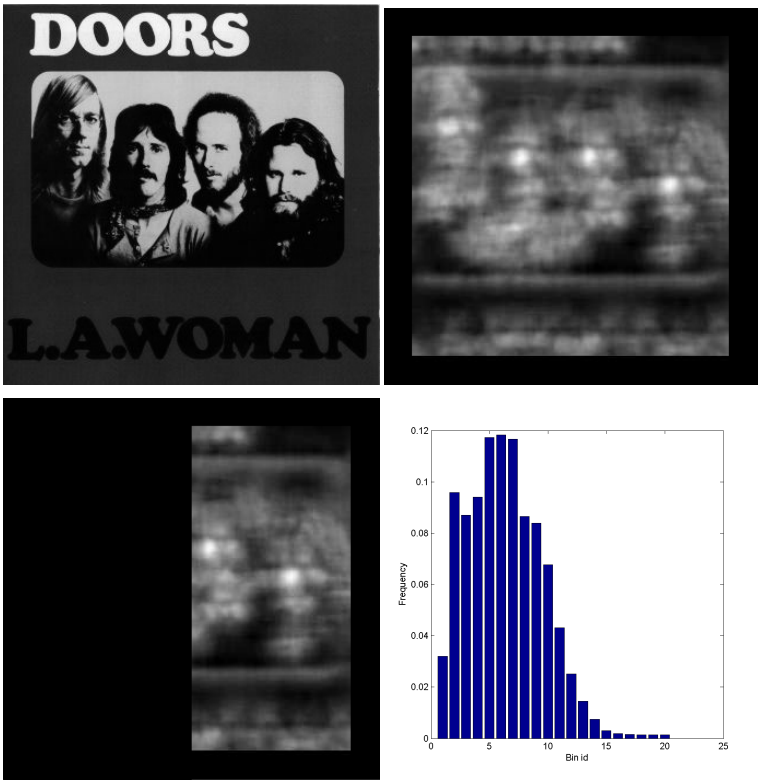


Figure 8.8: (row 1): Left: An image with 4 faces. Right: Result from the poor-classifier. row(2): Left: The masked response corresponding to the first dyadic region A_1. Right: The histogram on the region A_1.

Fig. 8.9 presents the instance localization problem in the language of channel communication. θ_i represents the location of the i^{th} target of interest, where $i = 1, \ldots, k$. $Y(\theta)$ is a k-sparse binary vector of length M^2. Each entry in Y corresponds to a pixel location in the image. Y_i at pixel location i is set to 1 if there is an instance of the object at this location as indicated by θ_i. The *source encoder* transforms the vector Y to a sequence of digits $Z_{1:N}$, which are the answers to the screening questions in our setting. $Z = AY$, where the coding or measurement matrix $A \in \{0,1\}^{N \times M^2}$, where N is the number of questions and M^2 is the size of the image. The noisy *channel* is analogous to the classifier

Figure 8.9: Coding Analogy for Object Localization

that finds the answers to the screening questions, more specifically this is the object counter in our face illustration, and $X_{1:N}$ are the noisy answers provided by the classifier. Finally the *decoder* finds an estimate of the location $\hat{Y}$ using the posterior computed from $X_{1:N}$ and the refinement algorithms PR algorithm (Alg. 4) or IPR algorithm (Alg. 5).

Setting up object localization as a channel communication problem offers significant benefits. This facilitates using ideas from classical coding theory to create new questions that augment the dyadic policy. Coding theory also offers proven performance bounds that helps us decide the minimum accuracy guarantees required from our machine learning classifiers.

In order to illustrate the viability of this idea, we now present a policy derived from Hamming code, which is a linear error correcting block code used in communication. A (p, q) Hamming code has code-word length p and data length q. Each message block, denoted by u consists of q information digits and $(p - q)$ parity bits. There are a total of 2^q distinct messages. The encoder transforms each input message u into a binary p-tuple v. This binary p-tuple v is referred to as the code vector for the message u. The parity bits for Hamming code are generated such that single bit errors can be detected and corrected.

We now show how the Hamming policy works by using a 4×4 image

grid example. The number of dyadic questions available for a 4×4 image is 4. We can use a $(7, 4)$ Hamming code to generate more screening regions for the 4×4 image. The 4-bit binary representation of all the 16 pixels in the image form the set of all messages. Each of these messages of length $q = 4$ are transformed using the Hamming encoder to generate 16 new code-words, each of length $p = 7$. This process gives use 3 extra questions compared to the dyadic policy. Note that query region A_n is obtained my setting to 1 all pixel indices in the 4×4 grid whose n^{th} significant bit is 1. Fig. 8.10 shows the 7 new basis images that can be used as the screening regions for our experiments. Note that Hamming codes include all the dyadic query regions, and is thus a superset of the dyadic policy.

Fig. 8.11 compares the performance of the PR algorithm (Alg. 4) for Hamming and dyadic policies for $k = 2$, $k = 3$ and $k = 10$ object instances for image sizes $\{8 \times 8, 16 \times 16, \ldots, 1024 \times 1024\}$. Our simulation experiments show that augmenting the screening query set using the Hamming policy provides better results than when using the dyadic policy alone.

8.4.1 Object detection: Hamming and Dyadic

We now use the Hamming policy to generate the screening regions and then employ the binary classifier to find the answers to the screening queries, which is then used to compute a posterior distribution on object instance locations. Table. 8.2 compares the performance of Hamming and dyadic policies for 96 images in the MIT+CMU face dataset. As was demonstrated in the simulation experiments in Fig. 8.11, Hamming shows slight improvement over dyadic for

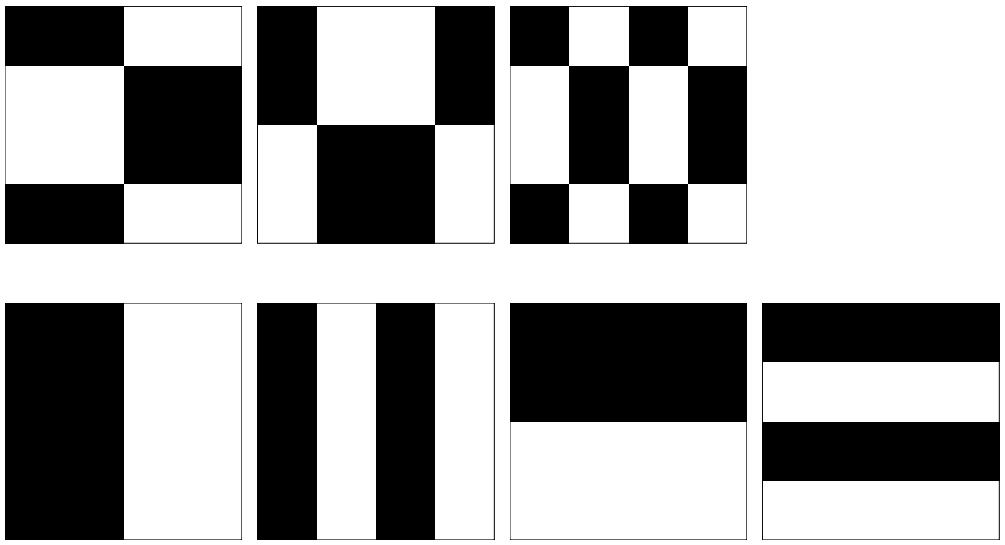


Figure 8.10: The queried regions generated using a $(7, 4)$ Hamming code for a 4×4 image grid.

Table 8.2: Results for Posterior Rank algorithm on a dataset with at most 4 faces. 96 images containing 115 faces in total. Hamming shows slight improvement over dyadic.

Policy	No. of Pixels	Oracle Calls	Oracle Calls per face
Hamming	56954880	606849	5276
Dyadic	56954880	732743	6371

face detection as well, which justifies further exploration into creating policies based on error correcting block codes.

(Note that the results presented in Table 8.2 uses a binary counter to find the answers to the dyadic queries.)

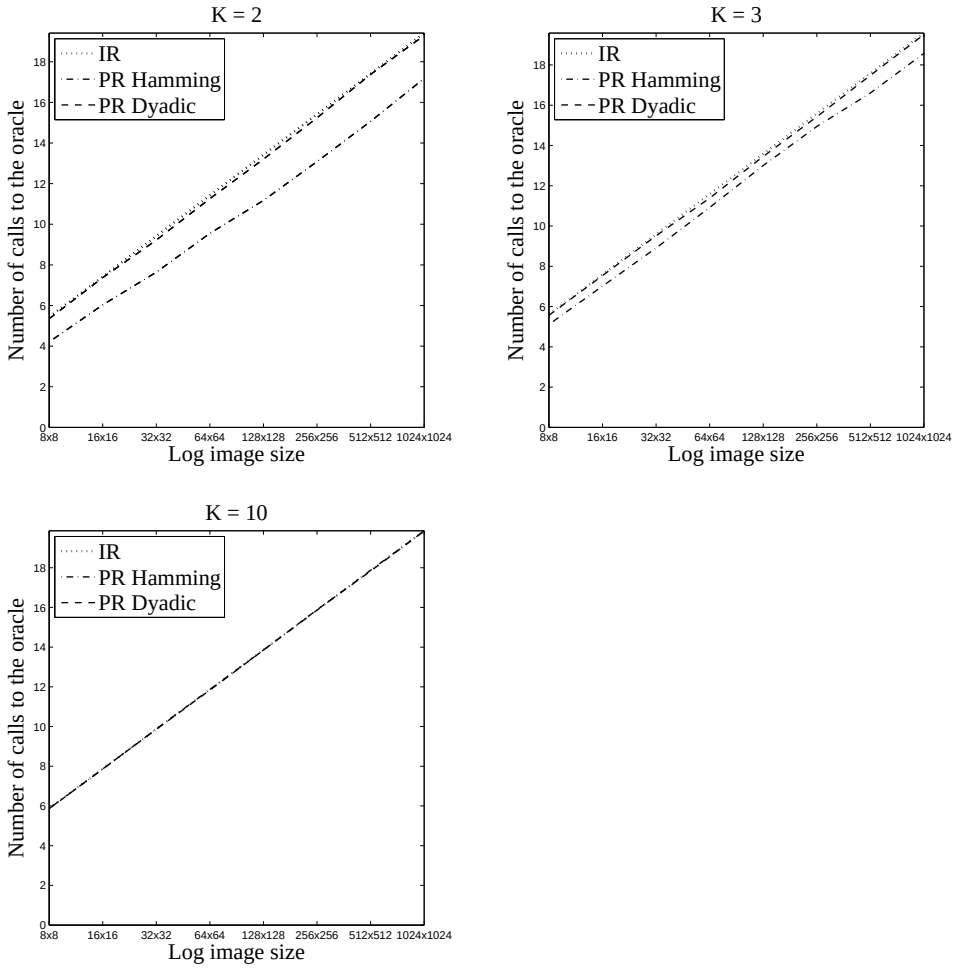


Figure 8.11: The mean number of calls to the oracle over 100 samples plotted against the image size for $k = 2$, $k = 3$ and $k = 10$ object instances using the PR algorithm (Alg. 4). A binary counter is used to get the answers to the screening questions. The curves compare the Dyadic and Hamming policies.

Chapter 9

Conclusion

9.1 Localization via Counting: Dyadic Policy and Cox Model

We now consolidate the counting algorithm presented in Part I with the dyadic localization algorithms from Part II, chapter 8. One of the key requirements of the dyadic policy is that it needs object counts from all query regions in order to compute the posterior distribution for object localization. Recall that the dyadic regions are comprised of horizontal and vertical stripes that span the input image, as shown in Figs. 8.2 and 7.3. These regions are obtained by recursively dividing the image into 2^n partitions at question n. As n increases, the strips get narrower, finally resulting in a screening region that is made up of regions that are just one pixel wide. even thought the localization algorithm is robust to a reasonable amount of noise in the subregion counts, the localization performance improves with count accuracy.

Estimating the counts from the poor-classifier response was a reasonable approach for our illustrative experiments in part II, section 8.3. Experiments

using binary counts (*i.e.* setting the region count to 1 if there is at least one object in the dyadic region and 0 otherwise) instead of the true counts for screening answers were promising as well (see 8.2). However, in order to leverage the full power of the dyadic localization algorithm, we need non-binary counters that work well. This is where the Cox counting framework comes in. The posterior intensity obtained after applying the Cox model can be used as is to find answers to the screening queries on any sub-region A_n. Note that according to the definition of a Poisson process, the sum of the intensity over a sub-image provides the expected number of instances within this sub-image and this modeling naturally fits in with the input requirement for the dyadic policy.

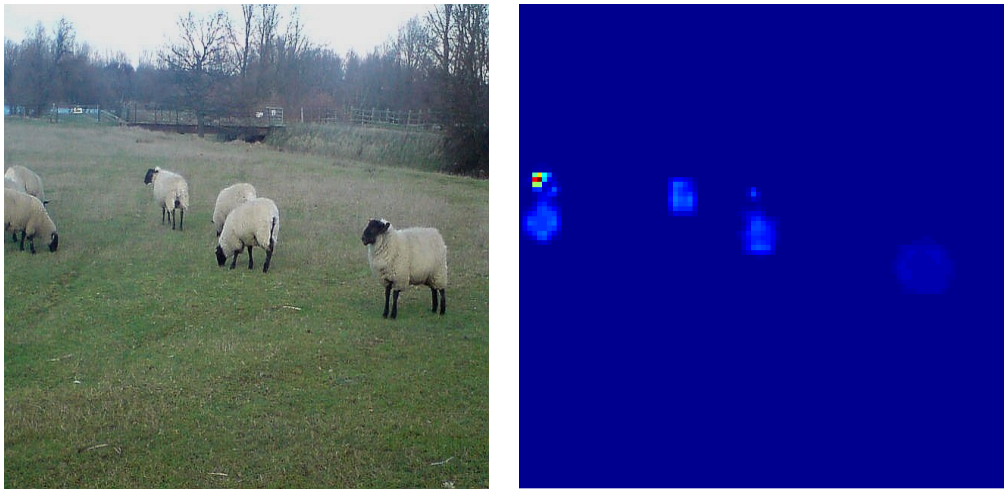


Figure 9.1: (left) Original Image from MS COCO, "sheep" category. (right) The Cox posterior intensity for the same image.

We now demonstrate this approach on images from MS-COCO [24] dataset.

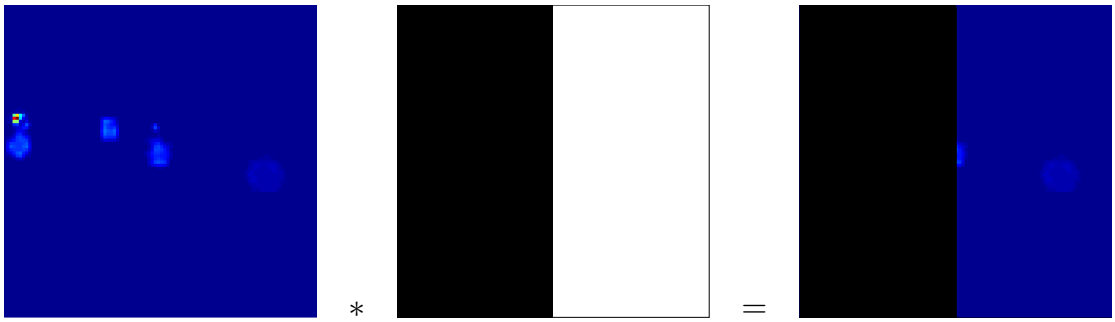


Figure 9.2: (left) The Cox posterior intensity for the sample image in fig. 9.1. (middle) The first dyadic mask (right) The masked Cox posterior intensity. The sum of the intensity in the masked region = 2, which is the number of sheep in that region.

First, we run the Cox counting algorithm to compute the Poisson posterior intensity over the input image (See Fig. 9.1). We then apply the dyadic basis images over the posterior intensity image and find the answers to all the screening questions by masking the regions that are not included in the query region. Fig. 9.2 illustrates this process for the first dyadic region. After adding up the values in the masked posterior, we use the counting regression parameters learned from the training data to map the sum to the estimated count. In this manner, we find the answers to all the dyadic questions. We then run the Posterior Rank (Alg. 4) and Iterated Posterior Rank (Alg. 5) algorithms for localization: *i.e.* compute the dyadic posterior distribution, rank the locations in the order of how likely it is to find an object at that location, and run the oracle according to this order until all the instances are found. The results are shown in Figs. 9.3 and 9.4. Both figures show the original input image with ground truth bounding box annotation and the dyadic posterior image with the detections shown as red rectangles. For IPR, the posterior is recomputed after each object is located, Fig. 9.4 shows the

posterior image at each stage of the IPR algorithm. These results show that the *dyadic policy* used in conjunction with the Cox counting model represents a novel means of localizing objects efficiently.

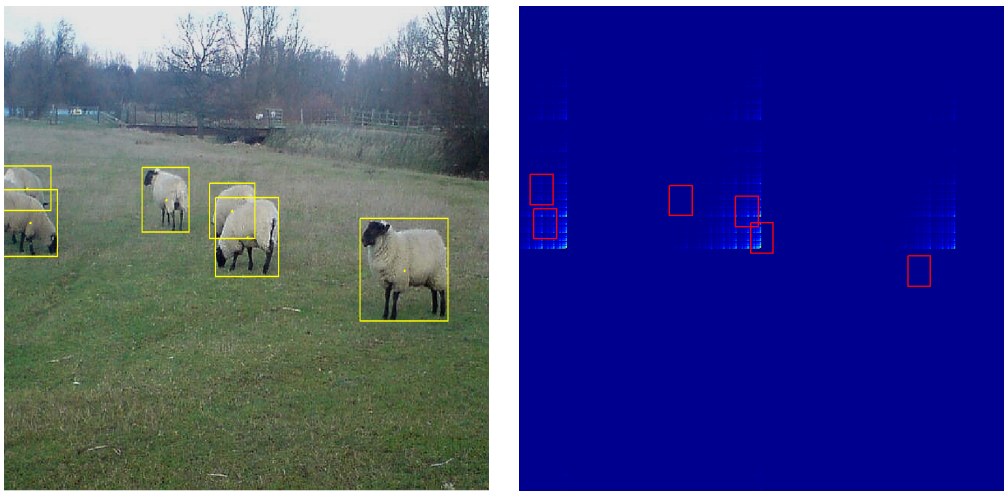


Figure 9.3: (left) Original Image with the ground truth annotation shown as yellow bounding boxes. (right) The posterior found by the Posterior Rank Algorithm 4. The locations of the objects detected are indicated as red rectangles in the image on the right.

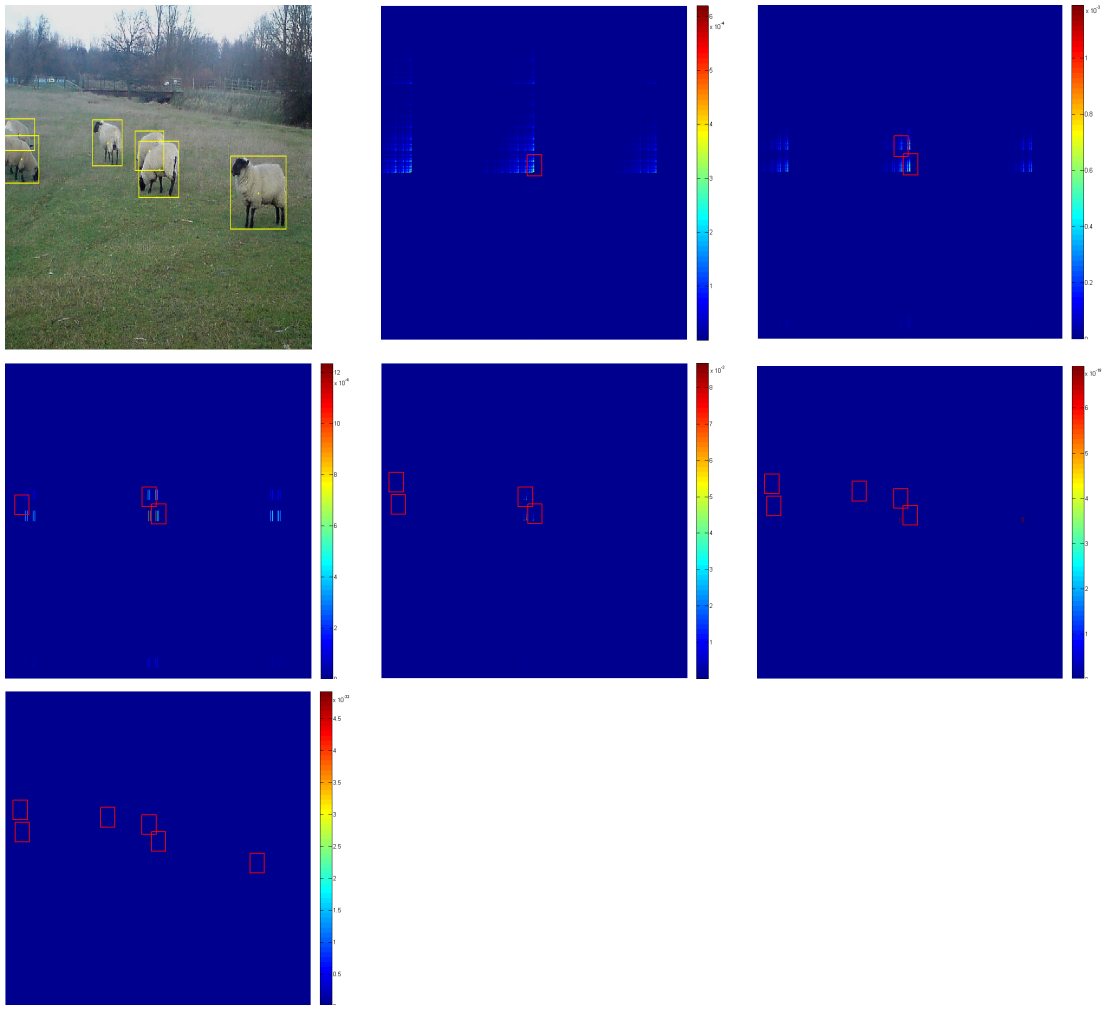


Figure 9.4: Original Image with the ground truth annotation shown on the top-left. The rest of the images show the posterior at each stage of IPR algorithm 4. The location of the face detected at each stage is shown as a red rectangle.

9.2 Conclusions

The goal of this dissertation was to develop algorithms for visual object counting and localization, deriving inspiration from classical techniques in Bayesian statistics. In part I of this thesis, we presented an efficient framework for estimating the number of instances of any object category of interest from images of natural scenes, based on doubly stochastic Poisson (Cox) processes. Evaluation using both synthetic and real data demonstrates that our algorithm scales well and improves upon the state-of-the-art in visual counting. In addition to counting, our Cox algorithm also allows for soft localization of objects. Since the location information is retained in the posterior intensity, this approach can be extended for precise object localization in the form of tight bounding boxes around the object. Moreover, this algorithm can be applied as-is to higher dimensional data.

We also extended this model to improve the effectiveness of pre-existing object counting and detection networks that use *objectness* scores over sub-windows. Specifically, we convert such region proposals to a well defined Poisson intensity by inserting a doubly stochastic Poisson (Cox) model into an already built and trained network. This idea can be applied to any network that uses region proposals in one of the intermediate layers. We demonstrated this approach on *Faster RCNN* (F-RCNN), which is one of the best performing object detection networks. Our experiments show that enhancing F-RCNN's region proposals using our Cox algorithm improves upon its counting results.

We also conducted preliminary object *detection* experiments using the same approach, *i.e.* updating the region proposal scores based on the Poisson

posterior intensity and continuing the F-RCNN flow as is for detection. Our initial results show an improvement in detection performance as well when compared to raw F-RCNN. The approach is powerful in the sense that we achieved this without any re-training or elaborate hyper-parameter tuning of the base network - it was more of a single step *plug and play*.

In part II of this thesis, we tackled the problem of localizing several targets simultaneously, based on object counts from image sub-regions. We derived a close-to-optimal policy within a Bayesian framework, using the expected entropy of the posterior as a value function. We then empirically evaluated this policy on a computer vision problem for the localization of several instances of the same object in image data, showing dramatic performance increases compared to a baseline method. We have also demonstrated that our algorithms are robust to a reasonable level of noise.

Finally, we used the Cox counter as the basis for performing *object localization by counting*, consolidating and building upon the ideas presented in parts I [30] and II [31, 32]. The preliminary consolidation experiments presented in this thesis show that there is promise in these ideas.

It is significant that these results have been achieved by adapting classical tools from Bayesian statistics in combination with state-of-the-art classifiers in computer vision. Furthermore, they were obtained without using additional data or computing power, instead focusing on algorithmic improvements. If incorporated into existing models, this approach of optimizing an intermixing of traditional and modern techniques has the potential to further accelerate advances in object localization and counting.

Appendix A

Cox Processes for Counting

A.1 Method of Moments: Detailed Derivation

Let Ω be the image domain with size $|\Omega| = a \times b$, where a and b are respectively the number of rows and columns of pixels in the image. Let $N(\Omega)$ be the number of instances of the object within Ω. According to the model, $N(\Omega)|\lambda \sim \text{Poisson}\left(\int_\Omega \lambda(s)\, ds\right)$, where $\lambda(s) = \alpha g^2(s)$ and g is a Gaussian Process $g \sim GP(0, K)$ with,

$K(s_1, s_2) = \sigma^2 exp\left\{-\frac{1}{2l^2}\|s_1 - s_2\|^2\right\}$.

Assume that we have m samples $N_1(\Omega) \ldots N_m(\Omega)$ over the same domain Ω, or over domains of the same size $|\Omega|$. Note that,

$$\begin{aligned} E[N(\Omega)] &= E[E[N(\Omega)|\lambda]] \\ &= E\left[\int_\Omega \lambda(s)\, ds\right] \\ &= \int_\Omega E[\lambda(s)] ds \\ &= \int_\Omega E[\alpha g^2(s)] ds \end{aligned}$$

$$= \alpha \int_{\Omega} K(s,s)\, ds$$

$$= \alpha \int_{\Omega} \sigma^2 ds$$

$$= \alpha\sigma^2 |\Omega| \tag{A.1}$$

The variance $V[N(\Omega)]$ can be written as,

$$V[N(\Omega)] = V[E[N(\Omega)|\lambda]] + E[V[N(\Omega)|\lambda]] \tag{A.2}$$

Using A.1, the second term of A.2 above simplifies easily to,

$$E[V[N(\Omega)|\lambda]] = E\left[\int_{\Omega} \lambda(s)\, ds\right] \tag{A.3}$$

$$= \alpha\sigma^2 |\Omega| \tag{A.4}$$

Let us now consider the first term of A.2,

$$V[E[N(\Omega)|\lambda]] = V\left[\int_{\Omega} \lambda(s)\, ds\right]$$

$$= E\left[\left(\int_{\Omega} \lambda(s)\, ds\right)^2\right] - E^2\left[\int_{\Omega} \lambda(s)\, ds\right]$$

$$= E\left[\left(\int_{\Omega} \alpha g^2(s)\, ds\right)^2\right] - \left(\alpha\sigma^2 |\Omega|\right)^2$$

$$= \alpha^2 \int_{\Omega}\int_{\Omega} E\left[g^2(s)g^2(t)\right] dsdt - \left(\alpha\sigma^2 |\Omega|\right)^2$$

$$= \alpha^2 \int_{\Omega}\int_{\Omega} 2cov^2(g(s), g(t)) + v[g(s)]v[g(t)]dsdt$$

$$- \left(\alpha\sigma^2 |\Omega|\right)^2$$

$$= 2\alpha^2\sigma^4 \left(\int_{s_1=0}^{a} \int_{t_1=0}^{a} exp\{-\frac{1}{l^2}(s_1 - t_1)^2\} ds_1 dt_1 \right)$$

$$\left(\int_{s_2=0}^{b} \int_{t_2=0}^{b} exp\{-\frac{1}{l^2}(s_2 - t_2)^2\} ds_2 dt_2 \right) \tag{A.5}$$

$$+ \alpha^2\sigma^4 |\Omega|^2 - \left(\alpha\sigma^2 |\Omega|\right)^2$$

In order to simplify the double integral in Eq. A.5, let us set $l^2 = 2l'^2$,

$$\begin{aligned}
\int_{t_1=0}^{a} \int_{s_1=0}^{a} exp\{-\frac{1}{l^2}(s_1 - t_1)^2\} ds_1 dt_1 &= \int_{t_1=0}^{a} \left(\int_{s_1=0}^{a} exp\{-\frac{1}{l^2}(s_1 - t_1)^2\} ds_1 \right) dt_1 \\
&= \int_{t_1=0}^{a} \left(\sqrt{2\pi} l' \int_{s_1=-\infty}^{\infty} \frac{1}{\sqrt{2\pi} l'} exp\{-\frac{1}{2l'^2}(s_1 - t_1)^2\} ds_1 \right) \\
&= \int_{t_1=0}^{a} \sqrt{2\pi} l' dt_1 \\
&= \sqrt{2\pi} a l' \qquad \text{(A.6)}
\end{aligned}$$

Using Eq. A.6, Eq. A.5 can be simplified as,

$$\begin{aligned}
V\left[E\left[N\left(\Omega\right)|\lambda\right]\right] &= 2\alpha^2\sigma^4 \sqrt{2\pi} a l' \sqrt{2\pi} b l' + \alpha^2\sigma^4 |\Omega|^2 - \left(\alpha\sigma^2 |\Omega|\right)^2 \\
&= 2\alpha^2\sigma^4 \pi a b l^2 \\
&= 2\alpha^2\sigma^4 \pi |\Omega| l^2 \qquad \text{(A.7)}
\end{aligned}$$

Combining Eq. A.7 and A.3, the variance A.2 can be expressed as,

$$V\left[N\left(\Omega\right)\right] = 2\alpha^2\sigma^4 \pi |\Omega| l^2 + \alpha\sigma^2 |\Omega| \tag{A.8}$$

The sample mean and variance are,

$$\bar{N} = \frac{1}{m}\sum_{i=1}^{m} N_i(\Omega) = \alpha\sigma^2 |\Omega|$$

$$S = \frac{1}{m}\sum_{i=m}^{m} (N_i(\Omega) - \bar{N})^2 = 2\alpha^2\sigma^4 2\pi abl^2 + \alpha\sigma^2 |\Omega|$$

Based on the method of moments, by equating the sample mean and variance with the theoretical mean and variance, we have the following two equations,

$$\bar{N} = E[N(\Omega)]$$

$$S = V[N(\Omega)]$$

From Eq. A.1 and Eq. A.8,

$$\bar{N} = \alpha\sigma^2 |\Omega|$$

$$S = 2\alpha^2\sigma^4 2\pi abl^2 + \alpha\sigma^2 |\Omega|$$

Solving for σ and l (set $\alpha = 1$),

$$\begin{aligned} \sigma &= \sqrt{\frac{\bar{N}}{|\Omega|}} \\ l &= \sqrt{\frac{|\Omega|\,(S - \bar{N})}{2\pi\bar{N}^2}} \end{aligned} \tag{A.9}$$

A.2 Concavity of the Counting Forward Model

Recall that $\lambda_m = \alpha\phi(g_m)$, with $\alpha > 0$. Choosing $p = 2$, we obtain for the model in Eq. (2.3),

$$\nabla_{g_m} lnp(y|g) = -4\beta(g_m - \sqrt{y_m})^3 \tag{A.10}$$

$$\nabla\nabla_{g_m} lnp(y|g) = -12\beta(g_m - \sqrt{y_m})^2 \tag{A.11}$$

Note that the last quantity is continuous and negative so that the matrix W is positive semi-definite.

Appendix B

Bayesian Multiple Target Localization

B.1 Expected Entropy

To support the proof of Theorem 2 and 3, we need the following lemma, which provides an expression for the expected entropy after additional questions. First of all, we introduce some notations. For any pair of random variables W, V, we define $H(W\|V)$ to be the random variable taking the value

$$-\int_{-\infty}^{\infty} p(w|v) \log p(w|v)\, dw \tag{B.1}$$

for each $V = v$, assuming the conditional density function $p(w|v)$ exists. And $H(W|V)$ is the formal conditional entropy.

In addition, for any random variables W, V, U, we define $I(W; V\|U)$ to be the random variable taking the value

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} p(w, v|u) \log \frac{p(w, v|u)}{p(w|u)p(v|u)}\, dv\, dw, \tag{B.2}$$

for each $U = u$, assuming the conditional density functions exist. And

$I(W;V|U)$ is the formal conditional mutual information.

Lemma 2. *Under any policy π, for all $n \geq 0$,*

$$E[H(p_{n+1})|X_{1:n}] = H(p_n) - I(\theta; X_{n+1}\|X_{1:n}). \tag{B.3}$$

Moreover,

$$E[H(p_N)] = H(p_0) - \sum_{n=0}^{N-1} I(\theta; X_{n+1}|X_{1:n}). \tag{B.4}$$

Proof. First of all, we prove the recursive relation (B.3). $H(p_n)$ is the entropy of the posterior distribution of θ, which is random through its dependence on the past history $X_{1:n}$, hence we can rewrite it as $H(p_n) = H(\theta\|X_{1:n})$. Similarly, $H(p_{n+1}) = H(\theta\|X_{1:n+1}) = H(\theta\|X_{1:n}, X_{n+1})$. Since all three terms in (B.3) are $\sigma(X_{1:n})$-measurable random variables, it suffices to prove (B.3) holds for any fixed history $X_{1:n} = x_{1:n}$, i.e.

$$E[H(\theta\|X_{1:n}, X_{n+1})|x_{1:n}] = H(\theta|x_{1:n}) - I(\theta; X_{n+1}|x_{1:n}). \tag{B.5}$$

Using information theoretic arguments, we have

$$E[H(\theta\|X_{1:n}, X_{n+1})|x_{1:n}] \tag{B.6a}$$

$$= H(\theta|X_{n+1}, x_{1:n}) \tag{B.6b}$$

$$= H(\theta, X_{n+1}|x_{1:n}) - H(X_{n+1}|x_{1:n}) \tag{B.6c}$$

$$= H(\theta|x_{1:n}) + H(X_{n+1}|\theta, x_{1:n}) - H(X_{n+1}|x_{1:n}) \tag{B.6d}$$

$$= H(\theta|x_{1:n}) - I(\theta; X_{n+1}|x_{1:n}) \tag{B.6e}$$

where (B.6b) comes from the definition of conditional entropy and (B.6c), (B.6d) come from the chain rule for conditional entropy. (B.6e) holds due to the relationship between entropy and mutual information. This proves (B.5).

Now, taking the expectation over $X_{1:n}$ on both sides of (B.3),

$$E\left[E[H(p_{n+1})|X_{1:n}]\right] = E[H(p_n)] - E\left[H(X_{n+1}\|X_{1:n})\right]. \tag{B.7}$$

Note that $E\left[E[H(p_{n+1})|X_{1:n}]\right] = E[H(p_{n+1})]$ by the iterated conditioning property of conditional expectation. Moreover, $E\left[I(\theta; X_{n+1}\|X_{1:n})\right] = I(\theta; X_{n+1}|X_{1:n})$ according to the definition of conditional entropy in (B.2). Hence, (B.7) is equivalent to

$$E[H(p_{n+1})] = E[H(p_n)] - I(\theta; X_{n+1}|X_{1:n}). \tag{B.8}$$

Applying (B.8) iteratively for $n = N-1, \ldots, 0$, we obtain (B.4), which concludes the proof. □

B.2 Proof of Theorem 2

Proof. According to Lemma 2, it suffices to prove that $I(\theta; X_{n+1}|X_{1:n}) \leq C_k \leq \log(k+1)$ for all $n \geq 0$ under any valid policy π. Since X_{n+1} depends on θ only through Z_{n+1}, we have

$$\begin{aligned} I(\theta; X_{n+1}|X_{1:n}) &= I(Z_{n+1}; X_{n+1}|X_{1:n}) \\ &= H(X_{n+1}|X_{1:n}) - H(X_{n+1}|Z_{n+1}, X_{1:n}) \\ &= H(X_{n+1}|X_{1:n}) - H(X_{n+1}|Z_{n+1}). \end{aligned} \tag{B.9}$$

Also, we have

$$H(X_{n+1}|X_{1:n}) = H\left(\sum_{z=0}^{k} \pi(z) f(\cdot|z)\right), \tag{B.10}$$

where $\pi(z)$ denotes the marginal distribution of Z_{n+1} and $f(\cdot|z)$ is the conditional probability density (mass) function of X_{n+1} given Z_{n+1}. Moreover,

$$H(X_{n+1}|Z_{n+1}) = \sum_{z=0}^{k} \pi(z) H\left(f(\cdot|z)\right). \tag{B.11}$$

Substituting (B.10) and (B.11) into (B.9) gives

$$\begin{aligned}
&I(\theta; X_{n+1}|X_{1:n}) \\
&= H\left(\sum_{z=0}^{k} \pi(z) f(\cdot|z)\right) - \sum_{z=0}^{k} \pi(z) H\left(f(\cdot|z)\right) \\
&\leq \sup_{q} H\left(\sum_{z=0}^{k} q(z) f(\cdot|z)\right) - \sum_{z=0}^{k} q(z) H\left(f(\cdot|z)\right) \\
&= C_k,
\end{aligned} \tag{B.12}$$

where $q(\cdot)$ is any probability mass function over $\{0, \ldots, k\}$.

To see the second inequality, we note that the channel capacity C_k is bounded from above by the capacity of a noiseless channel, i.e.

$$C_k \leq I(Z_{n+1}, Z_{n+1}) = H(Z_{n+1}). \tag{B.13}$$

Since Z_{n+1} is a discrete random variable over $\{0, \ldots, k\}$, the maximum possible value for the entropy $H(Z_{n+1})$ is obtained when X_{n+1} has a uniform distribution over $\{0, \ldots, k\}$. Therefore, $H(Z_{n+1}) \leq \log(k+1)$, which completes the proof. □

B.3 Proof of Theorem 3

Proof. We first show that the noiseless answers $Z_{1:n}$ are iid under the dyadic policy. Let $U_{i,j}$ be iid Bernoulli$(1/2)$ random variables and let V_i be iid Uniform$(0, 2^{-N-1})$. Then $T_i := \sum_{j=1}^{N} 2^{-j} U_{i,j} + V_i$ are iid Uniform$(0,1)$. By the inversion method for simulation, $Q(T_i) = F_0^{-1}(T_i)$ provides a random variable that has cdf F_0, and so is equal in distribution to θ_i. Because T_i is independent across i, and θ_i is independent across i, the vector $(Q(T_i) : i = 1, \ldots, k)$ is equal in distribution to θ. Each Z_n, considered as a function of θ, is equal in distribution to $U_{1,n} + ... + U_{k,n}$. Moreover, the vector $(Z_n : n = 1, \ldots, N)$ is equal in distribution to the vector $(U_{1,n} + ... + U_{k,n} : n = 1, \ldots, N)$, which is iid across n. Thus, $Z_{1:N}$ are iid.

Now, according to Lemma 2, it suffices to prove that under the dyadic policy, $I(\theta; X_{n+1}|X_{1:n}) = D_k$ for all $n \geq 0$. Under the dyadic policy, the noiseless answer $Z_{n+1} \sim \text{Bin}\left(k, \frac{1}{2}\right)$ and is independent of the previous history $X_{1:n}$ (this is a consequence of the independence of Z_{n+1} from $Z_{1:n}$ shown above). Hence, the marginal distribution function of Z_{n+1} is $\pi(z) = \binom{k}{z}\frac{1}{2^k}$. The remainder of the proof is similar to the proof of Theorem 1. □

B.4 Approximation Ratio

Now, we prove the claim made in Sect. 7.4.2, Corollary 1 regarding the approximation ratio in the noiseless case.

Lemma 3. $H(\text{Bin}(k, \frac{1}{2}))/\log(k+1) \geq \frac{1}{2}$.

Proof. $H(\text{Bin}(k, \frac{1}{2})) = H(\sum_{i=1}^{k} B_i)$, where B_i are iid Bernoulli($\frac{1}{2}$). By Theorem 1 in [87], (but expressing entropy in base 2 instead of base e),

$$2^{2H(\text{Bin}(k,\frac{1}{2}))} \geq k2^{2H(B_1)} = 4k.$$

This implies that $H(\text{Bin}(k, \frac{1}{2})) \geq \frac{1}{2}\log(4k)$ and

$$\frac{H(\text{Bin}(k, \frac{1}{2}))}{\log(k+1)} \geq \frac{1}{2}\frac{\log(4k)}{\log(k+1)} \geq \frac{1}{2}.$$

□

B.5 IPR Simulation

Two alternative simulation results of the IPR algorithm are provided in Figures B.1 and B.2.

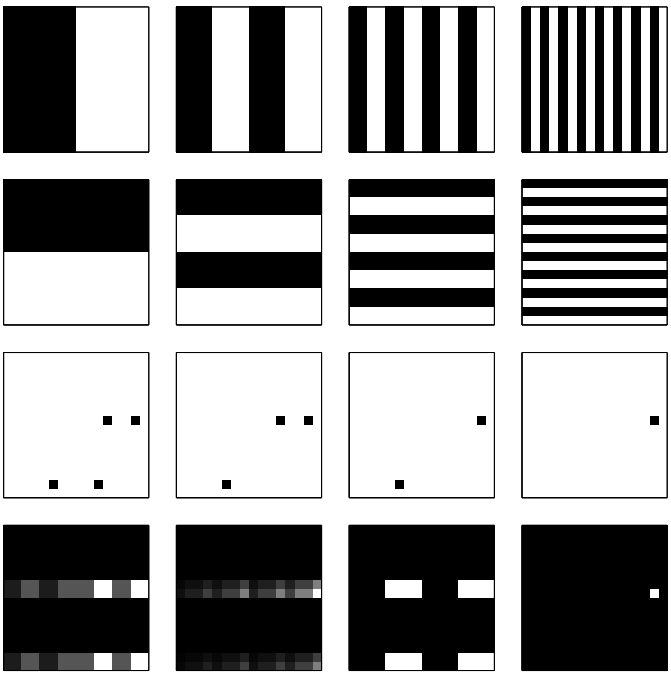


Figure B.1: Experiments in localization: (Top 2 lines) The queried regions under the dyadic policy for a 16×16 image shown in white. (3rd line) Example image with 4 instances of the object initially, one instance is found after each iteration of the IPR algorithm. (Last line) The corresponding posterior distribution after each iteration. The expected number of instances given the answers to the screening questions is proportional to the gray level; white indicating a large value.

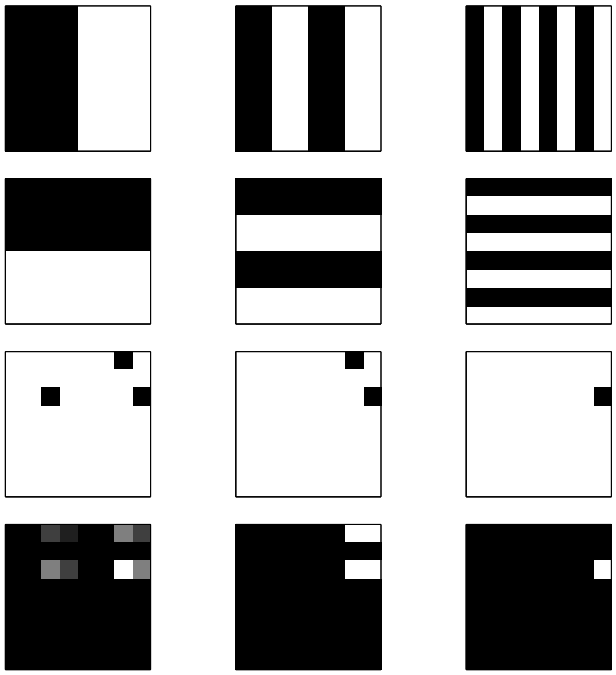


Figure B.2: Object localization: (Top 2 lines) The queried regions under the dyadic policy for a 8×8 image shown in white. (3rd line) Example image with 3 instances of the object initially, one instance is found after each iteration of the IPR algorithm. (Last line) The corresponding posterior distribution after each iteration. The expected number of instances given the answers to the screening questions is proportional to the grey level; white indicating a large value.

B.6 Detailed Implementation of the Entropy Pursuit Algorithm

A detailed implementation of the Entropy Pursuit (EP) algorithm is presented below.

Algorithm 7 Implementation of EP Part 1

1: Obtain noisy observations $x_1, ..., x_N$.
2: Generate E_N.
3: Create matrix D with dimension $2^N \times |E_N|$. Each row in D corresponds to one pixel C, each element in the row represents the number of instances in C as per each $S \in E_N$.
4: **for** $S_i \in E_N$ **do**
5: Update $Column\_i$ of D with the number of instances at each pixel C as per S_i;
6: **end for**
7: $m = 0, D^{(0)} = D, E_N^{(0)} = E_N$.
8: **for** each pixel C **do**
9: For each $u = 0, \ldots, k$, evaluate

$$\begin{aligned} &P(N(C) = u|x_{1:n}) \\ &= \frac{|\{S \in E_N^{(m)} : \text{codes for } C \text{ appear } u \text{ times in } S\}|}{|E_N^{(m)}|}; \end{aligned} \tag{B.14}$$

10: Evaluate

$$\begin{aligned} &H(N(C)|x_{1:n}) \\ &= -\sum_{u=o}^{k} P(N(C) = u|x_{1:n}) \log(P(N(C) = u|x_{1:n})); \end{aligned} \tag{B.15}$$

11: **end for**
12: **repeat**
13: $C^* = \arg\max_C \ H(N(C)|x_{1:n})$;
14: Query the oracle and obtain $Answer_{C^*}^{(m)} = Oracle(C^*)$;

Algorithm 8 Implementation of EP Part 2

15: **for** $S_i \in E_N^{(m)}$ **do**
16: **if** S_i is incompatible with $Answer_{C^*}^{(m)}$ **then**
17: Remove S_i from $E_N^{(m)}$;
18: Remove $Column_i$ from $D^{(m)}$;
19: **end if**
20: **end for**
21: m=m+1;
22: **until** $H(O_{C^*}|x_{1:n}) = 0$
23: The unique columns of $D^{(m)}$ give the estimated instances joint location .

B.7 Noise Models for the Dyadic Policy

B.7.1 Flip Noise

Now let us generalize our results to the noisy case. Suppose $\theta = (\theta_1, \ldots, \theta_K)$ are the targets to detect. Let $Z_n = \sum_{k=1}^{K} 1_{A_n}(\theta_k) = \sum_{k=1}^{K} z_k$ denote the true answer to the n^{th} question and let X_n denote the observed answer. Suppose the noise model takes the following form

$$X_n|Z_n = \sum_{k=1}^{K} y_k, \text{ where } x_k \sim \text{Bernoulli}\left((1-\epsilon)z_k + \epsilon(1-z_k)\right) \tag{B.16}$$

Now let us assume that the dyadic policy is employed and there are N questions. We have,

$$\begin{aligned} P(\theta_1 \in C|x_{1:N}) &= \sum_{z_{1:N}} P(X_1 \in C, z_{1:N}|x_{1:N}) \\ &= \sum_{z_{1:N}} P(X_1 \in C|z_{1:N}, x_{1:N}) P(z_{1:N}|x_{1:N}) \\ &= \sum_{z_{1:N}} P(X_1 \in C|z_{1:N}) w(z_{1:N}), \end{aligned} \tag{B.17}$$

where the weight

$$w(z_{1:N}) = P(z_{1:N}|x_{1:N}) = \frac{P(x_{1:N}|z_{1:N})P(z_{1:N})}{\sum_{z'_{1:N}} P(x_{1:N}|z'_{1:N})P(z'_{1:N})}. \tag{B.18}$$

$$w(z_{1:N}) \propto P(x_{1:N}|z_{1:N})P(z_{1:N}) \tag{B.19}$$

Also, recall from (7.35) that $P(\theta_i \in C|z_{1:N}) = \prod_{j=1}^{N} (\frac{z_j}{k})^{s_j}(1-\frac{z_j}{k})^{1-s_j}$

As a consequence,

$$P(\theta_i \in C|x_{1:N}) = \sum_{z_{1:N}} \prod_{j=1}^{N} (\frac{z_j}{k})^{s_j}(1-\frac{z_j}{k})^{1-s_j} w(z_{1:N}) \tag{B.20}$$

Let X_n and Z_n be the observed answer and the true answer to the n^{th} question respectively.

$X_n|Z_n \sim U_1 + ... + U_{z_n} + V_1 + ... + V_{K-z_n}$, where $U_i = Bernoulli(1-\epsilon)$ and $V_i = Bernoulli(\epsilon)$.

$\Longrightarrow$

$X_n|Z_n \sim U + V$, where $U = Binomial(z_n, 1-\epsilon)$ and $V = Binomial(K - z_n, \epsilon)$

We therefore have,

$$P(X_n = x_n|z_n) = \sum_{j=0}^{y_n} P(U = j)P(V = x_n - j) \tag{B.21}$$

Note,

$$x_n - j \leq K - z_n$$

$$j \geq x_n + z_n - K \tag{B.22}$$

$$\text{and } j \leq z_n$$

Therefore,

$$
\begin{aligned}
P(X_n = y_n|z_n) &= \sum_{j=max(0,x_1+z_1-K)}^{min(x_n,z_n)} P(U=j)P(V=x_n-j) \\
&= \sum_j \binom{z_n}{j}(1-\epsilon)^j\epsilon^{(z_n-j)}\binom{K-z_n}{x_n-j}\epsilon^{(x_n-j)}(1-\epsilon)^{(K-z_n-x_n+j)} \\
&= \sum_j \binom{z_n}{j}\binom{K-z_n}{x_n-j}(1-\epsilon)^{(K-z_n-x_n+2j)}\epsilon^{(z_n+x_n-2j)},
\end{aligned}
\tag{B.23}
$$

$$
P(X_{1:N}|Z_{1:N}) = \prod_{n=1}^{N} P(x_n|z_n) \tag{B.24}
$$

$$
P(Z_{1:N}) = \prod_{n=1}^{N} \binom{K}{z_n} 2^{-K} \tag{B.25}
$$

B.7.1.1 Simulation to compute the weights

Evaluating (B.19) is computationally demanding as there are $(K+1)^N$ terms in the sum. However, note that the weights decrease to zero exponentially with the distance between $x_{1:N}$ and $z_{1:N}$. As a consequence, we prepare a simulation aimed at estimating (B.19) by computing only some of the weights $w(z_{1:N})$. Here are the steps in the simulation.

- Observe noisy $x_{1:N}$
- Generate a set of $z_{1:N}$ values from the observed answers by changing one answer in the sequence $x_{1:N}$ at a time. We choose to change the observed answer by adding and subtracting one, while making sure that the values remain in the set $[0, K]$. This means there are at most

$m = 2N + 1$ possibilities.

- Calculate the relative weights, $w(z_{1:N})$ for each $z_{1:N}$ using (B.19) .
- Normalize the weights,

$$w_i = \frac{w_i}{\sum_{j=1}^{m} w_j} \tag{B.26}$$

B.7.2 Uniform Noise

Now assume we have the following noise model.

$$X_n|Z_n = \begin{cases} Z_n, \text{ with probability } 1-\epsilon, \\ \{0,1,\ldots,K\}\backslash Z_n, \text{ with probability } \frac{\epsilon}{K}. \end{cases} \tag{B.27}$$

Then we have

$$P(\theta_1 \in C|B_N) = \sum_{z_{1:N}\in\{0,\ldots,K\}^N} P(\theta_1 \in C|z_{1:N})w(z_{1:N}), \tag{B.28}$$

where we define $w(z_{1:N}) = P(z_{1:N}|B_N)$.

Now, define $\mathcal{D}_j = \{z_{1:N} \in \{0,\ldots,K\}^N | \sum_{n=1}^{N} \mathbb{1}_{\{z_n \neq x_n\}} = j\}$, for $j = 0,1,\ldots,N$. Namely, $\mathcal{D}_j$ is the subset where $z_{1:N}$ differs from $x_{1:N}$ in exactly j number of components. Then $\mathcal{D}_j's$ form a partition of $\{0,1,\ldots,K\}^N$. Then, we can rewrite (B.28) as

$$\begin{aligned} P(\theta_1 \in C|B_N) &= \sum_{j=0}^{N} \sum_{z_{1:N}\in\mathcal{D}_j} P(\theta_1 \in C|z_{1:N}) \left(\frac{\epsilon}{K}\right)^j (1-\epsilon)^{N-j} \\ &= \sum_{j=0}^{N} \left(\frac{\epsilon}{K}\right)^j (1-\epsilon)^{N-j} \sum_{z_{1:N}\in\mathcal{D}_j} P(\theta_1 \in C|z_{1:N}). \end{aligned} \tag{B.29}$$

Note that we have a closed form formula for the weight $w(z_{1:N}) = \left(\frac{\epsilon}{K}\right)^j (1-$

$\epsilon)^{N-j}$ for $z_{1:N} \in \mathcal{D}_j$, only depending on the information–for which $\mathcal{D}_j$ $z_{1:N}$ belong.

Moreover, there is a fast way to compute the sum $\sum_{z_{1:N}\in\mathcal{D}_j} P(\theta_1 \in C|z_{1:N})$ by applying Eq. (7.34). First of all, let us define the index set: $\mathcal{J}_C = \{n = 1, \ldots, N|C \subset A_j\}$. In other words, $\mathcal{J}_C$ is the index set such that $s_n = 1$. Define $p^*(C) = P(\theta_1 \in C|z_{1:N} = x_{1:N})$.

Consider the case where $\mathcal{D}_j = \mathcal{D}_1$. For each sequence $z_{1:N} \in \mathcal{D}_1$, we have $z_{1:N}$ differs from $x_{1:N}$ in one component whose index i is either in $\mathcal{J}_C$ or in $\mathcal{J}_C^c$. Using Eq. (7.34), we have,

$$P(\theta_1 \in C|z_{1:N}) = \begin{cases} p^*(C)\frac{z_i}{x_i}, \text{ if } i \in \mathcal{J}_C, \\ p^*(C)\frac{K-z_i}{K-x_i}, \text{ if } i \in \mathcal{J}_C^c. \end{cases} \tag{B.30}$$

Now we can compute the summation

$$\begin{aligned} \sum_{z_{1:N}\in\mathcal{D}_1} P(\theta_1 \in C|z_{1:N}) &= p^*(C)\left(\sum_{i\in\mathcal{J}_C}\left(\sum_{k=1}^{K}\frac{k}{x_i} - 1\right) + \sum_{i\in\mathcal{J}_C^c}\left(\sum_{k=1}^{K}\frac{k}{K-x_i} - 1\right)\right) \\ &= p^*(C)\left(\frac{K^2+K-2}{2}|\mathcal{J}_C|\sum_{i\in\mathcal{J}_C}\frac{1}{x_i} + \frac{K^2+K-2}{2}|\mathcal{J}_C^c|\sum_{i\in\mathcal{J}_C^c}\frac{1}{K-x_i}\right) \\ &= p^*(C)\frac{K^2+K-2}{2}\left(|\mathcal{J}_C|\sum_{i\in\mathcal{J}_C}\frac{1}{x_i} + |\mathcal{J}_C^c|\sum_{i\in\mathcal{J}_C^c}\frac{1}{K-x_i}\right), \end{aligned} \tag{B.31}$$

where the -1 term in the summation above is to deduct the case $z_i = x_i$.

For the equation above to be valid, we must have $p^*(C) \neq 0$, i.e. $x_i \neq 0$ for all $i \in \mathcal{J}_C$ and $x_i \neq K$ for all $i \in \mathcal{J}_C^c$. Now consider the case $x_i = 0$ for some $i \in \mathcal{J}_C$ or $x_i = K$ for some $i \in \mathcal{J}_c^c$. Then we define a new sequence $(x_i')_{i=1}^N$

such that

$$x_i' = \begin{cases} \frac{1}{2}, \text{ if } x_i = 0, i \in \mathcal{J}_C, \\ K - \frac{1}{2}, \text{ if } x_i = K, i \in \mathcal{J}_C^c, \\ x_i, \text{ otherwise.} \end{cases} \tag{B.32}$$

Define $p^*(C) = P(\theta_1 \in C | z_{1:N} = x'_{1:N}) \neq 0$. Note that if $\sum_{n=1}^{N} \mathbb{1}_{\{x_n' \neq x_n\}} \geq 2$ then $\sum_{z_{1:N} \in \mathcal{D}_1} P(\theta_1 \in C | z_{1:N}) = 0$. Hence, we only consider the case where (x_i') differs from (x_i) in only one component, i^*. Then we have for either $i^* \in \mathcal{J}_C$ or $i^* \in \mathcal{J}_C^c$,

$$\sum_{z_{1:N} \in \mathcal{D}_1} P(\theta_1 \in C | z_{1:N}) = p^*(C)(K^2 + K). \tag{B.33}$$

B.8 Unknown Number of Objects

Now we consider the case where the number of targets, K, is unknown. Assume $K \in \mathcal{K}$ follows a prior distribution p_K. We aim at computing the expected number of targets in each bin C after observing a sequence of noisy answers $X_{1:n} = x_{1:n}$. Let $N(C)$ denote the number of targets in C. The conditional expectation of $N(C)$ can be computed as

$$E_n[N(C)] = \sum_{k \in \mathcal{K}} E_n[N(C)|K = k] p_n(K = k) = \sum_{k \in \mathcal{K}} k p(\theta_1 \in C | x_{1:n}, K = k) p_n(K = k). \tag{B.34}$$

Since the term $p(\theta_1 \in C | x_{1:n}, K = k)$ can be obtained from Section B.7.1, we only need to consider the term $p_n(K = k)$. Note that

$$p_n(K = k) = \sum_{z_{1:n} \in \{0, \ldots, k\}^n} p_n(K = k | z_{1:n}) p_n(z_{1:n}). \tag{B.35}$$

Moreover,

$$p_n(K=k|z_{1:n}) = \frac{p(z_{1:n}|K=k)p_K(k)}{p(z_{1:n})}, \tag{B.36a}$$

$$p_n(z_{1:n}) = p(z_{1:n}|x_{1:n}) = \frac{p(x_{1:n}|z_{1:n})p(z_{1:n})}{p(x_{1:n})}, \tag{B.36b}$$

where we noticed that $p(x_{1:n})$ is a constant as the $X_{1:n} = x_{1:n}$ is given.

Also, note that

$$p(x_{1:n}|z_{1:n}) = \sum_{k' \in \mathcal{K}} p(x_{1:n}|z_{1:n}, K=k')p(K=k'|z_{1:n}) \tag{B.37}$$

where $p(x_{1:n}|z_{1:n}, K=k')$ can be obtained from Section B.7.1, and $p(K=k'|z_{1:n})$ is immediate after we choose the first question to be the entire unit interval $(0,1]$.

Above all,

$$p_n(K=k) \propto \sum_{z_{1:n} \in \{0,\dots,k\}^n} \left(p(z_{1:n}|K=k)p_K(k) \sum_{k' \in \mathcal{K}} p(x_{1:n}|z_{1:n}, K=k')p(K=k'|z_{1:n}) \right). \tag{B.38}$$

B.8.1 Unknown K case with expected answers conditional on noisy observations

Assume the number of targets K is unknown and we want to compute the posterior likelihood $P(\theta_1 \in C|B_N)$, and thus obtaining the expected number of targets in C,$E_n[N(C)]$. According to (B.34), a crucial step is to compute the posterior distribution of K. We first compute the conditional probability as

follows.

$$
\begin{aligned}
P(K=k|X_1,\ldots,X_n) &\propto P(X_1,\ldots,X_n|K=k)P_K(k) \\
&= \prod_{j=1}^{n} P(X_j|K=k)P_K(k) \\
&= \left(\prod_{j=1}^{n}\sum_{z_j=0}^{k} P(X_j|Z_j=z_j,K=k)P(Z_j=z_j|K=k)\right)P_K(k) \\
&= \left(\prod_{j=1}^{n}\sum_{z_j=0}^{k} \binom{k}{z_j}2^{-k}P(X_j|Z_j=z_j,K=k)\right)P_K(k).
\end{aligned}
\tag{B.39}
$$

Note that the term $P(X_j|Z_j=z_j,K=k)$ is specified by the noisy model such as the uniform noise model given in (B.27).

Now consider a given history B_n. The posterior distribution of K denoted by $P_n(K=k)$ can be computed using (B.39). Then, the expected number of objects in C can be obtained by

$$
E_n[N(C)] = \sum_{k=0}^{\infty} P(\theta_1 \in C|B_n,K=k)P_n(K=k), \tag{B.40}
$$

where $P(\theta_1 \in C|B_n,K=k)$ is the posterior likelihood in the known K case, which we can compute using $P(\theta_i \in C|B_N) = P(\theta_1 \in C|Z_{1:N}=e_{1:N})$ (See Eq. 7.38)

Bibliography

[1] John Whalen, Charles R Gallistel, and Rochel Gelman. Nonverbal counting in humans: The psychophysics of number representation. *Psychological Science*, 10(2):130–137, 1999.

[2] Manuela Piazza and Véronique Izard. How humans count: numerosity and the parietal cortex. *The neuroscientist*, 15(3):261–273, 2009.

[3] Lars Chittka and Karl Geiger. Can honey bees count landmarks? *Animal Behaviour*, 49(1):159–164, 1995.

[4] Raghavendra Gadagkar. Can animals count? *Current Science*, 68(12):1180–1182, 1995.

[5] Hans J Gross. To bee or not to bee, this is the questionâĂę the inborn numerical competence of humans and honeybees: The inborn numerical competence of humans and honeybees. *Communicative & integrative biology*, 4(5):594–597, 2011.

[6] A. Merchan-Perez, J. Rodriguez, L. Alonso-Nanclares, A. Schertel, and J. DeFelipe. Counting synapses using fib/sem microscopy: A true revolution for ultrastructural volume reconstruction. *Frontiers in Neuroanatomy*, 3(18), 2009.

[7] Graham J Edgar, Neville S Barrett, and Alastair J Morton. Biases associated with the use of underwater visual census techniques to quantify the density and size-structure of fish populations. *Journal of Experimental Marine Biology and Ecology*, 308(2):269–290, 2004.

[8] G Merz. Counting elephants (loxodonta africana cyclotis) in tropical rain forests with particular reference to the tai national park, ivory coast. *African Journal of Ecology*, 24(2):61–68, 1986.

[9] Chris Hopkinson, Laura Chasmer, Colin Young-Pow, and Paul Treitz. Assessing forest metrics with a ground-based scanning lidar. *Canadian Journal of Forest Research*, 34(3):573–583, 2004.

[10] P Gilbertson and AD Bradshaw. Tree survival in cities: the extent and nature of the problem. *Arboricultural Journal*, 9(2):131–142, 1985.

[11] Kaggle competition noaa fisheries steller sea lion population count. Accessed: 2017-04-28.

[12] Yang Cong, Haifeng Gong, Song-Chun Zhu, and Yandong Tang. Flow mosaicking: Real-time pedestrian counting without scene-specific learning. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1093–1100. ieee, 2009.

[13] Antoni B Chan, Zhang-Sheng John Liang, and Nuno Vasconcelos. Privacy preserving crowd monitoring: Counting people without people models or tracking. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–7. IEEE, 2008.

[14] Cong Zhang, Hongsheng Li, Xiaogang Wang, and Xiaokang Yang. Cross-scene crowd counting via deep convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 833–841, 2015.

[15] Haroon Idrees, Imran Saleemi, Cody Seibert, and Mubarak Shah. Multi-source multi-scale counting in extremely dense crowd images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2547–2554, 2013.

[16] Weina Ge and Robert T Collins. Marked point processes for crowd counting. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2913–2920. IEEE, 2009.

[17] Min Li, Zhaoxiang Zhang, Kaiqi Huang, and Tieniu Tan. Estimating the number of people in crowded scenes by mid based foreground segmentation and head-shoulder detection. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1–4. IEEE, 2008.

[18] Ryan Greene-Roesel, Mara Chagas Diogenes, David R Ragland, and Luis Antonio Lindau. Effectiveness of a commercially available automated pedestrian counting device in urban environments: Comparison with manual counts. 2008.

[19] Thomas Moranduzzo and Farid Melgani. Automatic car counting method for unmanned aerial vehicle images. *IEEE Transactions on Geoscience and Remote Sensing*, 52(3):1635–1647, 2014.

[20] Chenda Liao and Prabir Barooah. An integrated approach to occupancy modeling and estimation in commercial buildings. In *American Control Conference (ACC), 2010*, pages 3130–3135. IEEE, 2010.

[21] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[22] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[23] J Deng, A Berg, S Satheesh, H Su, A Khosla, and L Fei-Fei. Ilsvrc-2012, 2012.

[24] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer, 2014.

[25] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders. Selective Search for Object Recognition. *International Journal of Computer Vision*, 104(2):154–171, 2013.

[26] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645, 2010.

[27] D. Mortlock. Astronomy: The age of the quasars. *Nature*, 514:43–44, 2009.

[28] K. Ali, F. Fleuret, D. Hasler, and P. Fua. A real-time deformable detector. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(2), 2012.

[29] D. Lee, K. Lee, W. Ho, and S. Lee. Target cell-specific involvement of presynaptic mitochondria in post-titanic potentiation at hippocampal mossy fiber synapse. *Journal of Neuroscience*, 27(50):13603–13613, 2007.

[30] Purnima Rajan, Yongming Ma, and Bruno Jedynak. Cox processes for counting by detection. *Journal of Mathematical Imaging and Vision*, pages 1–14, 9 2018.

[31] Purnima Rajan, Weidong Han, Raphael Sznitman, Peter Frazier, and Bruno Jedynak. Bayesian multiple target localization. In *International Conference on Machine Learning*, pages 1945–1953, 2015.

[32] Weidong Han, Purnima Rajan, Peter I Frazier, and Bruno M Jedynak. Bayesian group testing under sum observations: A parallelizable two-approximation for entropy loss. *IEEE Transactions on Information Theory*, 63(2):915–933, 2017.

[33] John Frank Charles Kingman. *Poisson processes*. Wiley Online Library, 1993.

[34] Baback Moghaddam and Alex Pentland. Probabilistic visual learning for object representation. *IEEE Transactions on pattern analysis and machine intelligence*, 19(7):696–710, 1997.

[35] Trung Thanh Pham, Tat-Jun Chin, Konrad Schindler, and David Suter. Interacting geometric priors for robust multimodel fitting. *IEEE transactions on image processing*, 23(10):4601–4610, 2014.

[36] Carl Edward Rasmussen. Gaussian processes for machine learning. 2006.

[37] David R Cox. Some statistical methods connected with series of events. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 129–164, 1955.

[38] Angelos Dassios and Ji-Wook Jang. Pricing of catastrophe reinsurance and derivatives using the cox process with shot noise intensity. *Finance and Stochastics*, 7(1):73–95, 2003.

[39] Seth Flaxman, Andrew Gordon Wilson, Daniel B Neill, Hannes Nickisch, and Alexander J Smola. Fast kronecker inference in gaussian processes with non-gaussian likelihoods. In *International Conference on Machine Learning*, volume 2015, 2015.

[40] David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[41] Yunus Saatçi. *Scalable inference for structured Gaussian process models*. PhD thesis, Citeseer, 2012.

[42] Victor Lempitsky and Andrew Zisserman. Learning to count objects in images. In *Advances in Neural Information Processing Systems*, pages 1324–1332, 2010.

[43] Luca Fiaschi, Ullrich Koethe, Rahul Nair, and Fred A Hamprecht. Learning to count with regression forest and structured labels. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 2685–2688. IEEE, 2012.

[44] Carlos Arteta, Victor Lempitsky, J Alison Noble, and Andrew Zisserman. Interactive object counting. In *European Conference on Computer Vision*, pages 504–518. Springer, 2014.

[45] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[46] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[47] Dan Kong, Douglas Gray, and Hai Tao. A viewpoint invariant approach for crowd counting. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 3, pages 1187–1190. IEEE, 2006.

[48] AN Marana, SA Velastin, LF Costa, and RA Lotufo. Estimation of crowd density using image processing. In *Image Processing for Security Applications (Digest No.: 1997/074), IEE Colloquium on*, pages 11–1. IET, 1997.

[49] Siu-Yeung Cho, Tommy WS Chow, and Chi-Tat Leung. A neural-based crowd estimation by hybrid global learning algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 29(4):535–541, 1999.

[50] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[51] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[52] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *European Conference on Computer Vision*, pages 346–361. Springer, 2014.

[53] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.

[54] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. *arXiv preprint arXiv:1612.08242*, 2016.

[55] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[56] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448, 2015.

[57] Trung T Pham, Seyed Hamid Rezatofighi, Ian Reid, and Tat-Jun Chin. Efficient point process inference for large-scale object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2837–2845, 2016.

[58] Florent Lafarge, Xavier Descombes, et al. Geometric feature extraction by a multimarked point process. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1597–1609, 2010.

[59] Yannick Verdie and Florent Lafarge. Detecting parametric objects in large scenes by monte carlo sampling. *International Journal of Computer Vision*, 106(1):57–75, 2014.

[60] Vishwanath A Sindagi and Vishal M Patel. A survey of recent advances in cnn-based single image crowd counting and density estimation. *Pattern Recognition Letters*, 2017.

[61] Chuan Wang, Hua Zhang, Liang Yang, Si Liu, and Xiaochun Cao. Deep people counting in extremely dense crowds. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 1299–1302. ACM, 2015.

[62] Elad Walach and Lior Wolf. Learning to count with cnn boosting. In *European Conference on Computer Vision*, pages 660–676. Springer, 2016.

[63] Vishwanath A Sindagi and Vishal M Patel. Cnn-based cascaded multi-task learning of high-level prior and density estimation for crowd counting. In *Advanced Video and Signal Based Surveillance (AVSS), 2017 14th IEEE International Conference on*, pages 1–6. IEEE, 2017.

[64] Deepak Babu Sam, Shiv Surya, and R Venkatesh Babu. Switching convolutional neural network for crowd counting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, page 6, 2017.

[65] Daniel Onoro-Rubio and Roberto J López-Sastre. Towards perspective-free object counting with deep learning. In *European Conference on Computer Vision*, pages 615–629. Springer, 2016.

[66] Sheldon M Ross. Stochastic processes. 1996, 1996.

[67] Alexander Graham. *Kronecker products and matrix calculus with applications*. Courier Dover Publications, 2018.

[68] Huamin Zhang and Feng Ding. On the kronecker products and their applications. *Journal of Applied Mathematics*, 2013, 2013.

[69] B. Jedynak, P. Frazier, and R. Sznitman. Twenty questions with noise: Bayes optimal policies for entropy loss. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1:114–136, 2012.

[70] M. Horstein. Sequential transmission using noiseless feedback. *IEEE Transactions on Information Theory*, 9(3):136–143, Jul 1963.

[71] R. Castro and R. Nowak. Active sensing and learning. *Foundations and Applications of Sensor Management*, 2007.

[72] R. Waeber, P. I. Frazier, and S. G. Henderson. Bisection search with noisy responses. *SIAM J. Control and Optimization*, 51(3):2261–2279, 2013.

[73] T. Tsiligkaridis, B. M. Sadler, and A. O. Hero. Collaborative 20 questions for target localization. *CoRR*, abs/1306.1922, 2013.

[74] M V Burnashev and K S Zigangirov. On One Problem of Observation Control. *Problemy Peredachi Informatsii*, 11(3):44–52, 1975.

[75] D. Du and F. Hwang. *Combinatorial Group Testing and its Applications*. World Scientific Pub Co, 2000.

[76] D. Stinson, T. Trung, and R. Wei. Secure flameproof codes, key distribution patterns, group testing algorithms and related structures. *Journal of Statistical Planning and Inference*, 86:595–617, 2000.

[77] D. Eppstein, M Goodrich, and D Hirschberg. Improved combinatorial group testing algorithms for real-world problem sizes. *SIAM Journal on Computing*, 36:1360–1375, 2007.

[78] N. Harvey, M. Patrascu, Y. Wen, S. Yekhanin, and V. Chan. Non-adaptive fault diagnosis for all-optical networks via combinatorial group testing on graphs. In *IEEE International Conference on Computer Communications*, pages 697–705, 2007.

[79] E. Porat and A. Rothschild. *Altomata, Languages and Programming*, volume 5125, chapter Explicit Non-adaptive Combinatorial Group Testing Schemes, pages 748–759. Springer, Berlin Heidelberg, 2008.

[80] S. Kauffman. *At Home in the Universe: The Search for the Laws of Self-Organization and Complexity*. Oxford University Press, 1996.

[81] J. Buzas and G. Warrington. Optimized random chemistry. Technical Report 1302.2895, ArXiv e-prints, 2013.

[82] F. Chung, R. Graham, and T. Leighton. Guessing secrets. *The Electronic Journal of Combinatorics*, 8(13), 2001.

[83] R. Sznitman and B. Jedynak. Active testing for face detection and localization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(10):1914–1920, 2010.

[84] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[85] R. Sznitman, R. Richa, R. Taylor, B. Jedynak, and G. Hager. Unified detection and tracking of instruments during retina microsurgery. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(5):1263–1273, 2013.

[86] R. Sznitman, A. Lucchi, P. Frazier, B. Jedynak, and P. Fua. An optimal policy for target localization with application to electron microscopy. In *Proceedings of the 30th International Conference on Machine Learning*, pages 1–9, 2013.

[87] P Harremoes, C Vignat, et al. An entropy power inequality for the binomial family. *JIPAM. J. Inequal. Pure Appl. Math*, 4(5), 2003.

Vita

Purnima Rajan received her B-Tech in Electronics and Communication engineering from College of Engineering, Trivandrum, University of Kerala; and M.Sc. in Computer Science from Johns Hopkins University. She went on to complete her Ph.D. at the Department of Computer Science at Johns Hopkins University, where her research interests included computer vision, machine learning, and statistical methods applied to image analysis.

Prior to commencing her graduate studies, she worked extensively in the medical software industry, spending a total of 8 years at Siemens Medical Solutions, Siemens Oncology and Siemens Molecular Imaging. Towards the end of her Ph.D. studies, she interned at Philips Research North America in their health-care division. She will shortly be joining Clear Guide Medical, a technology company focused on the development of medical devices,

applying computer vision techniques to ultrasound-guided interventions.